Simple low cost causal discovery using mutual information and domain knowledge

Joseph, Adrian

Simple low cost Causal Discovery using Mutual Information and Domain Knowledge

Adrian Joseph

Submitted for the degree of Doctor of Philosophy

Queen Mary, University of London

2011

This thesis examines causal discovery within datasets, in particular observational datasets where normal experimental manipulation is not possible. A number of machine learning techniques are examined in relation to their use of knowledge and the insights they can provide regarding the situation under study. Their use of prior knowledge and the causal knowledge produced by the learners are examined. Current causal learning algorithms are discussed in terms of their strengths and limitations. The main contribution of the thesis is a new causal learner LUMIN that operates with a polynomial time complexity in both the number of variables and records examined. It makes no prior assumptions about the form of the relationships and is capable of making extensive use of available domain information. This learner is compared to a number of current learning algorithms and it is shown to be competitive with them.

Publications

Joseph A., Fenton N. F., Neil M. (2006). Predicting football results using Bayesian nets and other machine learning techniques Knowledge-Based Systems Volume 19, Issue 7, November 2006, Pages 544-553. Elsevier B.V.

Contents

List of Figures

List of Tables

List of Algorithms

Acknowledgements

I would like to acknowledge the patience of my supervisor, and the support of my wife and mother without which this work would never have existed. I would also like to thank Duncan and Thomas whose helpful comments have assisted me in improving this thesis.

Chapter 1

Introduction

This thesis is about learning causal relationships. The main research hypothesis is that it is possible to learn about causal relationships in observational datasets, those datasets where changes cannot be made to the system being observed, making use of existing domain knowledge and without many of the traditional limitations imposed by causal learning systems, and further to do so with a polynomial time complexity in both the number of variables and number of records in the dataset. This is demonstrated by the LUMIN algorithm introduced in this thesis. The initial research was directed at how to best incorporate domain knowledge into a variety of learning methods, and its converse how to extract domain knowledge from what those learners discovered. However, over time it became apparent that while many learners can provide accurate predictions and aid in the understanding of systems underlying the data, what is ultimately needed is a detailed explanation of the interactions in the system that produce the data. Such knowledge provides for a complete and accurate prediction of the behaviour of the system being studied. While such a complete picture is beyond what could be achieved in a limited research exercise, the simpler task of identifying the causal relationships within the data was chosen as a useful step on the longer journey of a more complete understanding of the relationships contained in the data. The first two chapters of the thesis largely represent the earlier work as does the published paper in Appendix A. The remaining chapters build on this information, but do so in the context of what is necessary to learn causal relationships from data, which is both very much more general in scope and very much more specific in focus than the earlier chapters.

1.1 Learning

Learning can be defined as: To gain knowledge, comprehension, or mastery of *something* through experience or study. In general we understand learning in human terms. We gain some understanding, knowledge or skill we previously lacked. Playing chess, driving a car and solving mathematical problems are all tasks we recognise as requiring learning. There are things we learn in ways that are not normally associated with learning, like understanding what we see or balancing when walking. These activities require learning, even though our brains and bodies have the required physiology. Another from of learning is associative learning, this allows otherwise unrelated events to be paired. Pavlov [Pavlov 1927], showed how unrelated events can be paired though experience to produce a response to an otherwise neutral stimulus.

Learning makes it possible to improve our performance at a wide variety of tasks. However, improving our performance at a task does not imply that we have learnt something. Increasing muscle strength and endurance can improve performance for some tasks without requiring the learning of any knowledge or skill. However, in the context of Machine Learning, ML, improved performance at a task is a useful metric when measuring the effect and success of learning.

1.2 Machine Learning

Like human learning ML covers a variety of different tasks. Many different methods have been devised to allow machines to learn various types of knowledge. The only unifying aspect is that they all learn something. So, how do we define a system which learns? A reasonable definition, borrowed from [Mitchell 1997], would be that a system is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T as measured by P, improves with experience E. This is a fairly broad definition which includes all *normal* learning systems, and covers others not generally thought of as learning systems, such as database query systems. However, it is a useful definition as it highlights a number of the key features involved in a learning system.

With any learning system we require a task to learn, some experiences or knowledge to learn from, and some method of determining how well we have learnt the task. The task will effect the choices available for the other aspects of the learning system. There are a number of forms of learning experience available: rote learning; trial and error; by example; and by deduction to name a few. The type of experience available will vary with the task. A chess playing program

could be given tutorial examples of chess positions together with some rating for the position. Such a program could also gain experience in a trial and error fashion by playing other programs or other instances of itself. A program to learn the best treatment for a particular disease would, at least in most modern societies, not be able to learn from trial and error. Thus it would be restricted to specified training examples and or historical data.

A common method used to understand the ML process is to model it as a search task [Wilson 1970]. In this model we have some hypothesis space and we search through this space for the hypothesis that best matches the available data. This model allows many common search techniques to be adapted to the problem of learning. However, there are general problems with ML that may not occur in other search problems. The exact form of the required hypothesis is not usually known. So, it is possible that the required target function, the function[1] we are trying to learn, cannot be represented within our hypothesis space.

ML can have a number of problems related to the available data. The data we have to learn from may be noisy and there may not be much data available. We have to assume that the data we have are representative of the problem as a whole. These problems require various techniques to try and minimise any detrimental effect on the learning task. Additionally, analysis of the problems and techniques can help to quantify the impact of noisy or limited data on the accuracy with which the target function will likely be learnt. ML techniques generally focus on producing this target function, and not on providing a theory of the underlying situation. The techniques also vary in how much use they make of existing information about the situation under study.

1.3 Knowledge and Experience

In the context of people, learning is generally taken to cover explicit knowledge and general understanding. We might judge a person's ability to solve a particular mathematical problem in terms of their explicit knowledge of the required techniques. However, our hypothetical problem solver might know a number of different techniques which might be able to solve the given problem. The choice of which to use might be based not on any explicit knowledge of their applicability to the given problem, but on experience with what are perceived to be similar problems, see cased-based learning section 2.5. People often approach problem solving with a mixture of explicit knowledge and understanding gained from experience. In ML a similar set of issues

[1] Function is a misnomer, but commonly used in the ML field.

exist. There are a number of different ML techniques and the choice of which is likely to be the best technique for a given situation requires an understanding of the techniques themselves, what is required from the learner, and what data is available for it to learn from.

1.4 Understanding and Predicting

We learn because by doing so we are able to do something better than we could previously. At the simplest level learning some information would allow us to answer a question on that information. Learning also allows us to understand things we previously did not. An example of this would be learning how a plane flies by changing the air pressure above and below its wings.

Learning is not an all or nothing process, someone can learn to drive, how to operate a car, without necessarily having any knowledge about how a car works. What they learn is that pressing the accelerator makes the car go faster without appreciating how the engine works or what pressing the accelerator does to the fuel feed to the engine. In essence they learn to understand the effect of their actions without understanding the mechanism underlying that effect. This separation between predicting the effects of an action and understanding the mechanisms by which that effect takes place is common in the context of ML techniques. If a situation is well understood then predictions can be made about the effect of changes. It is harder to gain understanding of the underlying mechanisms of a system by having a good, but opaque, predictor of that system's performance.

1.5 The Use of Prior Knowledge in Learning

Knowledge[2] is used at all stages in the learning process. In the simplest terms we always try to build on what we already know, and to use our existing knowledge to both define and simplify the new learning task, this approach is clear in both case-based learning, section 2.5, and rule-based learning, section 2.6. In people it can been seen that knowledge is an aid to learning. Methods of reasoning such as deduction and induction allow us to extend what is already known. Problem solving techniques, like dividing a difficult problem into smaller sub problems, allow complex situations to be understood. People are also good at spotting patterns, thus a new problem might be seen as being similar to a previously solved problem and the previously successful technique reused. In ML there are specific ways we can use existing knowledge.

[2]See section 3.1 for a brief discussion on the nature of knowledge.

In most situations when learning is required there exists some initial information or speculation as to the underlying situation under study. Initially there must be some trigger, often an observation, which points out some lack of knowledge or understanding. Then the problem to be solved, or area of ignorance to be reduced will be defined. This is really the first use of knowledge in the learning process, identifying what is not known. Once the target of the learning is identified then those features which are felt most likely to effect the target can be identified. This is a use of knowledge, since potentially any feature could effect the target function.

Selection of likely key features is a necessary first step in the ML process. In general ML techniques perform poorly when given a large number of irrelevant features, see section 2.13. So, a balance needs to be achieved which includes, what is hoped to be, a sufficient set of features to define the target function with few irrelevant features. With this information the learning process can begin.

Learning relies on some form of reasoning, there are two common forms of reasoning, deductive and inductive. Deductive reasoning starts with a set of premises, or rules, and a process which uses those premises possibly along with other knowledge to generate valid conclusions. Thus if the set of premises is true, then the conclusion will be true. An example of deductive reasoning is:

1. All men are mortal – the premise
2. Socrates is a man – a related fact
3. Therefore, Socrates is mortal – the conclusion

Inductive reasoning starts from a finite collection of specific observations and a process which evaluates the degree of support these observations give to a conclusion. Unlike deductive reasoning, inductive reasoning suggests the truth of the conclusion, but does not ensure it. An example of inductive reasoning is:

1. All the cats I've seen have tails – the observations
2. All cats have tails – the (false) conclusion

Deductive and inductive reasoning are closely related [Jevons 1874]:

> Induction is, in fact, the inverse operation of deduction, and cannot be conceived to exist without the corresponding operation, so that the question of relative importance cannot arise.
>
> – William Stanley Jevons

We will examine the use of prior knowledge in them separately below.

1.5.1 Use of Prior Knowledge with Deductive Reasoning

Deductive learning is the process of taking what is already known and using deductive reasoning to deduce *new* knowledge. In deductive learning our initial knowledge is usually in the form of true statements and the combination of the representation of the knowledge and the deduction rules determine what can be learnt. The majority of this thesis examines inductive learning, however, deductive learning in the form of explanation-based learning is examined in section 2.11. Deductive learning can form part of the learning process in other learners, such as Inductive Logic Programming, section 2.7 and case-based learning, see section 2.5, but it depends on the specific implementation, these learners are mostly inductive learners.

1.5.2 Use of Prior Knowledge with Inductive Reasoning

Inductive learning is the process of taking data and some rule(s) for determining relationship(s) between them, and using inductive reasoning to determine new relationships within the data. If we define the learning process as a search problem, [Wilson 1970], then potentially knowledge can be used in all stages of the search. Choosing the features to use and the type of learner defines the hypothesis space through which we will search for the target function.

The starting point of the search is important as learning, like many search processes, can become trapped in local maxima, in this case a locally most likely hypothesis. Using knowledge to choose a starting point of the search, the initial hypothesis, close to the likely final hypothesis, will help to optimise the learnt hypothesis.

In inductive learning we attempt to move through the hypothesis space towards the target function from our current hypothesis. Knowledge of the likely target function will allow us to choose appropriate steps to take. Knowledge of both the hypothesis space and the problem domain will allow us to choose a suitable test to determine if the most likely hypothesis has been reached.

1.6 Causality

The idea of causality is a common one and while most people have an intuitive understanding of what it is, there no single accepted method of defining it. This is a rather strange state of affairs when you consider that all of modern science is based on the idea of causality, if events could happen without a specific cause then science would be a worthless pursuit. We discuss the problem of defining causality in more detail in chapter 3.

For this thesis we assume a deterministic world view, one in which changes in the state of a system (at a time $t + \delta t, \delta t > 0$) occur due to the action of previous events (at a time t). The event at time t is the cause of the change in state at time $t + \delta t$. No state changes happen without a cause, and the cause must precede the effect it generates. This is more restrictive with respect to interventions than Rubin's definition [Rubin 1989], "X is a cause for Y if an external agent interfering only with X can affect Y", as it imposes an additional timing constraint. This world view is suited to the pursuit of science. Science then becomes the process of understanding the cause and effect relationships in any situation, both qualitatively and quantitatively.

1.7 Document Structure

This document is organised into the following chapters. Chapter 1 introduces the motivation for the research. Chapter 2 is a review of some current learning techniques in the context of this research. Chapter 3 is a look at what we mean by causality and at how we attempt to define it and at the meaning of probability values used in determining causal relationships. We also examine the relationship between logical implication and causal relationships. Chapter 4 examines some existing methods of learning causal networks. Chapter 5 introduces the LUMIN learner and discusses aspects of its design. Chapter 6 examines three well known causal networks and new networks specifically designed to test the capabilities of the LUMIN learner. We investigate what is learnt by a number of different causal learners and compare their findings with those of the LUMIN learner. Chapter 7 examines how well we met our aims and looks at possible direction of future research and development of these ideas.

Chapter 2

Review of Machine Learning Techniques

In this chapter we start, in section 2.1, with a look at the history of ML and artificial intelligence in general. We then examine inductive learning, section 2.2, which forms the basis of many learners. Section 2.3 to 2.12 cover a selection of ML learners, examine how they work, the knowledge they produce, and what causal information can be obtained from them. We discuss whether the learner allows domain information to be included in the learning process in addition to the raw data, and whether domain information can be extracted from what is learnt. The last part of the chapter deals with general techniques that can be applied to many learners to improve their performance. Section 2.13 details Feature Subset Selection which restricts data to improve performance. Section 2.14 explains various ways in which labeled and unlabeled data can be used. It is sometimes possible to manipulate data to allow the discovery of otherwise hidden relationships. This is explained in section 2.15. Its useful to understand the accuracy of any learner in a given situation and section 2.16 explains methods which allow testing of the accuracy of a learner's predictions. Lastly it is possible to improve the performance of a learner by grouping a number of learners together on the same task. These *ensemble* techniques are explained in section 2.17.

2.1 A Brief History of Artificial Intelligence

Many cultures have developed the idea of an artificial intelligence, AI, usually contained within an inanimate object. Such ideas can be traced through Greek mythology, Hephaestus a black-smith who made many mechanical servants and incidentally the first woman, and Homer wrote

about mechanical "tripods" waiting on the gods at dinner, and later in the sixteenth century to rabbi Judah Loew Ben Bezalel who is said to have created a golem to defend the jews of the Prague Ghetto. In more modern fiction Mary Shelley's creation Frankenstein could be seen as artificial, although the parts of his body were not, but perhaps a better analogy would be the robot Gort from the 1951 film The Day the Earth Stood Still. These are examples of not merely artificial intelligence, but of artificial beings able not only to think, but to direct actions. Automata are perhaps more practical examples of less ambitious attempts at an artificial intelligence. Complex automata appear to have existed in ancient Greece. The island of Rhodes would seem to have been a centre for the production of such devices.

> The animated figures stand
> Adorning every public street
> And seem to breathe in stone, or
> move their marble feet.
>
> – a part of Pindar's seventh Olympic Ode

While none of these objects has survived, the Antikythera mechanism shows that sophisticated mechanical devices were being constructed at that time. Yet more sophisticated automata are referred to in Liezi (Lieh-tzu, c.4th C. BCE) a Chinese philosophical text. According to the text a mechanical engineer known as Yan Shi proudly presented King Mu of Zhou with a life-size, human-shaped figure. The figure could walk, sing, dance and apparently flirt. The latter activity caused it to be largely dismantled in an attempt to save its creator from the king's wrath. While a cat may look at a queen, its not a good idea, even for an automaton, to flirt with a king's ladies. The history of thinking machines has a chequered past, one of the best known examples is that of the Mechanical Turk. This was supposed to be a chess playing automaton and was exhibited as such by its owners from 1770 until 1894 when it was destroyed by fire. However, the truth was revealed in the early 1820's, it turned out that the device was a hoax with a human chess player hidden inside.

While the ideas around machine learning and artificial intelligence have been around in one from or another for a very long time, the modern approach to it using computers is often credited to Alan Turing[1]. Turing proved that a simple machine, what today is usually called a Turing ma-

[1]See `http://en.wikipedia.org/wiki/Alan_Turing`.

chine, but he called an a-machine, would be capable of performing any conceivable mathematical computation if it were representable as an algorithm [Turing 1937]. A similar, if somewhat more complex, proof was published by Alonzo Church in terms of λ-calculus. It is not uncommon to refer to the result as the Church-Turing thesis. This along with other discoveries in neurology, information theory and cybernetics led to a situation where it became possible to think seriously about constructing an artificial intelligence.

The modern field of artificial intelligence was largely founded in the summer of 1956 at a conference on the campus of Dartmouth College. The conference included John McCarthy, Marvin Minsky, Allen Newell and Herbert Simon who would become major contributors to the new field[2]. The aims of the original researchers were very ambitious and they had considerable initial success [Newell & Simon 1956, Bobrow 1964, Winograd 1971]. However, despite the initial success and a great deal of optimism there proved to be many difficult problems to be solved in the quest for an artificial intelligence. Research into artificial intelligence has been cyclical in that there have been periods of significant advances, the defeat of world chess champion Gary Kasparov, in 1997 by the Deep Blue program being a notable one, with lots of optimistic predictions for general successes, followed by periods of slow progress with general scepticism for any likely breakthroughs. Those times when AI research appeared to have stalled and there was little interest or funding for new AI is often referred to as an AI Winter[3]. One of the results of this cycle and more generally of the difficulties in aiming to produce an artificial intelligence, is the development of a number of sub-fields of research. Learning is one area that has been identified as a key element required for AI [Minsky 1961]. One of the key ideas of the early work was the use of heuristics.

> In the absence of an effective method guaranteeing the solution to a problem in a reasonable time, heuristics may guide a decision maker to a very satisfactory, if not necessarily optimal, solution.
>
> – Herbert Simon

As well as the early AI successes there were numerous failures, machine based language translation was originally thought to be a solvable problem in a moderate length of time. However,

[2] It was John McCarthy who coined the term Artificial Intelligence.

[3] See also `http://en.wikipedia.org/wiki/AI_winter`, `http://www.inf.ed.ac.uk/about/AIhistory.html` and [Russell & Norvig 1995].

it proved to be much more complex than first thought, requiring more knowledge, context sensitivity, and understanding of syntactic ambiguity. The phrase 'Time flies like an arrow; fruit flies like a banana', is sometimes used to demonstrate some of the problems, see [Burck 1965] page 62 for its use in a modified form. Another significant area of development was knowledge-based systems. The use of knowledge-based systems in limited domains demonstrated how computers could be useful in applying expert knowledge without the need for an expert to be present. DENDRAL was a system, developed by Edward Feigenbaum and Joshua Lederberg from 1965, which attempted to find the structure of organic compounds based on the output of mass spectrometry. It was in the context of DENDRAL that the knowledge principle, [Lenat & Feigenbaum 1991], was first stated:

> A system exhibits intelligent understanding and action at a high level of competence primarily because of the specific knowledge that it can bring to bear: the concepts, facts, representations, methods, models, metaphors, and heuristics about its domain of endeavor.
>
> – Douglas B. Lenat and Edward A. Feigenbaum

MYCIN [Shortliffe & Buchanan 1975, Buchanan & Shortliffe 1984], is another knowledge-based system which attempted to diagnose which bacteria were causing severe infections, such as bacteremia and meningitis, and to recommend antibiotics. In research conducted at the Stanford Medical School MYCIN was found to be better at suggesting an acceptable therapy than the infectious disease experts. An interesting part of the MYCIN system was that it supplied information on the certainty of its diagnosis. While its use of 'certainty factors' has been called into question, particularly in some of its underlying assumptions, it did avoid the need for experts to supply a large number of conditional probability estimates. One of the long term issues for knowledge-based systems is the amount of knowledge, particularly of an uncertain nature, which they can require to be supplied by experts. Like DENDRAL and MYCIN, PROSPECTOR [Hart et al 1978], was another expert system, in this instance to help with the analysis of the location of likely mineral deposits.

One of the changes that occurred in the mid to late 1970's and throughout the 1980's was the commercialisation of AI research. Most of the original research has been done at universities using government grants. However, with AI projects producing useful programs various com-

mercial interests, and a number of the researchers, realised there was commercial potential in AI work. As a result many of the then successful research teams were split as people left to join or start companies. Another change was a realisation by the military establishments of a number of countries that AI could be useful to them. An interesting area of research with commercial applications was speech recognition. The DRAGON program, [Baker 1975], is a good example of this kind of work.

The early 1970's also saw the emergence of AI systems based on Bayesian probability, that is probability analysis using Bayes' theorem, see section 2.9. The initial work was mostly with the Naïve Bayes classifier [Duda & Hart 1973], but with the development of efficient propagation algorithms [Pearl 1988], Bayesian networks have been used for many different tasks [Spiegelhalter et al 1989, Charniak & Goldman 1989, Fenton & Wang 2006]. The basic building blocks of artificial neural networks have been around for some time [Rosenblatt 1957]. However, work on understanding how to train such networks advanced significantly in the late 1960's [Minsky & Papert 1969] and a general purpose training algorithm was developed in the 1980's [Rumelhart et al 1986]. Artificial neural networks have been used for a number of different tasks [Pomerleau 1993, Bishop 1996, Caruana et al 1996], and a number of different learning algorithms have been developed [Simard et al 1992, Mitchell & Thrun 1993, Riedmiller & Braun 1993, Igel & Husken 2003].

The early 1990's saw a new type of learner enter the picture, the support vector machine [Boser, Guyon & Vapnik 1992]. These provide a quite different mechanism to most of the other systems and have proved to be flexible in their application [Joachims 2002, Eitrich and Lang 2006, Sun et al 2007].

While there are many other AI techniques and new ones will be developed, the original development from symbolic and rule-based systems, to knowledge-based and probabilistic, often Bayesian, systems through to neural networks and support vector machines, has to some extent been superseded by the development of multi-agent systems, both independent interacting agents [Panait & Luke 2005] and ensembles, see section 2.17. One of the most recent trends is for ML systems to make use of more than one standard technique. These composite systems attempt to use the strengths of each different technique to enhance their overall performance.

The ML landscape we see today with research in many different areas is the result of working to solve parts of the overall problem, and the developments that have arisen while striving towards

the ultimate goal of an artificial intelligence. A good introduction to the history of AI can be found in [McCorduck 2004].

2.2 Inductive Learning

Many machine learning techniques attempt to generalise from a number of specific examples, this is called inductive learning. All such learning techniques rely on the inductive learning hypothesis to validate their results.

> **Definition 1.** The **inductive learning hypothesis** is that any hypothesis found to approximate the target function well over a sufficiently large set of training examples will also approximate the target function well over other unobserved examples.

Another general and important consideration is that of inductive bias. All inductive learning systems must have an inductive bias. A learning system that makes no *a-priori* assumptions regarding the identity of the target function has no rational basis for classifying any unseen instances. The prior assumptions made by the learning system, both explicit and implicit, are called the inductive bias. A more formal definition taken from [Mitchell 1997] is:

> **Definition 2.** Consider a learning algorithm L for the set of instances X. Let c be an arbitrary function defined over X, and let $D_c = \{< x, c(x) >\}$ be an arbitrary set of training examples of c. Let $L(x_i, D_c)$ denote the classification assigned to the instance x_i by L after training on the data D_c. The **inductive bias** of L is any minimal set of assertions B such that for any target function c and corresponding training examples D_c
>
> $$(\forall x_i \in X)\,[(B \wedge D_c \wedge x_i) \vDash L(x_i, D_c)] \tag{2.1}$$
>
> where $\vDash$ is the logical entailment operator so $A \vDash B$ means B follows deductively from A.

Inductive bias can occur in a number of different ways: the representation used can bias the hypothesis space (e.g Candidate-Elimination algorithm [Mitchell 1977]); the search algorithm can introduce bias (e.g. the ID3 algorithm [Quinlan 1986]); the performance measure can introduce bias (e.g. information gain see section 2.3); the initial hypothesis can introduce bias (e.g. neural networks see section 2.8); *a-priori* decisions can introduce bias (e.g. Bayesian networks [Pearl 1988]).

Another common inductive learning issue is that of overfitting. Overfitting occurs when the learning algorithm learns to fit the training data so well that is has a detrimental effect on fitting other non-training instances. This can be defined as follows:

> **Definition 3.** Given a hypothesis space H, a hypothesis $h \in H$ is said to **overfit** the training data if there exists some alternative hypothesis $h' \in H$, such that h has a smaller error than h' over the training examples, but h' has a smaller error than h over the entire distribution of instances.

There are many techniques which can reduce the probability of overfitting the training data, these include: cross-validation of the hypothesis with non-training data, see section 2.16; statistical analysis to determine if further specialisation or generalisation of the hypothesis will increase its accuracy over the instance space rather than just over the training set (see [Quinlan 1986] for this applied to decision trees); and attempting to measure the complexity of the hypothesis[4] to stop when this is minimised. This last is an idea based on the minimum description length, MDL, principle. The MDL principle described in [Rissanen 1978], is based on the ideas introduced by Solomonoff's on inductive inference [Solomonoff 1964a, Solomonoff 1964b]. Cross-validation techniques involve partitioning the examples into a number of distinct subsets. The learning will then be performed a number of times, each time a different subset will be omitted from the training data, but will instead be used for validation. Using this method an *average* target function can be induced which will hopefully perform better on unseen instances than one learnt using all the examples as training data.

2.3 Decision Trees

Decision tree learning is a useful technique for learning discrete valued functions. It is generally robust to noise in the data, can cope with missing attributes and allows for disjunctive representations. The tree structure produced by a decision tree learner can be converted to rules of the form

IF condition <and condition>... THEN conclusion

which are easier to read than an unmodified tree. A decision tree learner will produce a tree structure starting with a root node and extending down through any number of branch nodes,

[4] This is partly dependent on the representation being used.

finally ending in leaf nodes. Decision trees classify an instance by sorting them down the tree from the root node to a leaf node which supplies the classification for the instance.

Example	Sky	AirTemp	Humidity	Wind	Water	Forecast	EnjoySport
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes

Table 2.1: Positive and Negative Examples for the Target Concept EnjoySport

Using the EnjoySport example taken from [Mitchell 1997], given the data in table 2.1 a decision tree for the EnjoySport problem can be constructed as shown in figure 2.1. This is a very simple tree and is not the only possible tree which could be built from the four examples given in the EnjoySport problem. An alternative tree, figure 2.2, would also satisfy the learning requirement.

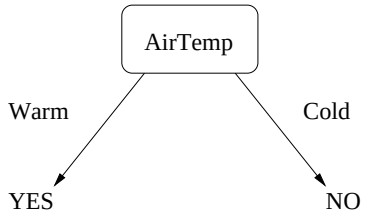


Figure 2.1: A Decision Tree Representation of the EnjoySport Problem

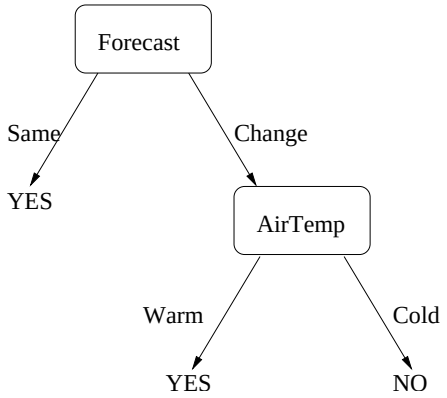


Figure 2.2: Alternative Decision Tree for EnjoySport

The two different trees demonstrate the importance of attribute choice when building a tree. Decision tree building algorithms, such as ID3 [Quinlan 1986] and C4.5 [Quinlan 1993], execute a top down greedy search through the hypothesis space. The choice of the best attribute when building a decision tree is a subject for research, what is required is to find the attribute that best determines the target function. This choice should produce the shortest and simplest tree. This is,

in effect, looking for an MDL bias. A statistical property, called information gain, measures how well a given attribute separates the training examples according to the target classification. This is defined using a measure from information theory called entropy. The entropy of a collection is a measure of the purity of the collection with respect to some classification.

Definition 4. Let S be a collection of instances of a discrete random variable, X, with possible values $\{x_1, x_2, \ldots, x_n\}$. Then the **entropy** of X in S, usually referred to as $H(X)$, is defined as:

$$H(X) := -\sum_{i=1}^{n} p(X = x_i) \log_2 p(X = x_i) \tag{2.2}$$

where $p(X = x_i)$ is the probability that an instance of variable X has the value x_i, we will usually simply write this as $p(x_i)$.

Given a set S, containing positive and negative examples of some target concept, the entropy of S relative to this boolean classification is:

$$H_{bool}(S) = -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus} \tag{2.3}$$

where $p_{\oplus}$ is the proportion of positive examples in S and $p_{\ominus}$ is the proportion of negative examples in S. The entropy of a set with respect to some classification ranges between 0, when all instances in the set are positive or negative, and 1 when an equal proportion of the instances in the set are positive and negative. The use of logarithm to base 2 is not related to the boolean nature of the example, but to the use in information theory of binary encoding. Information gain can be defined when choosing to partition a set using a particular attribute in terms of the reduction of the entropy of the set.

Definition 5. Given a set of examples S whose items contain attributes A, let $value(x, a)$, where $x \in S$, define the value of a specific example x for attribute $a \in A$. The **information gain**, $IG(S, A)$ of an attribute A, relative to a set of examples S, is defined as:

$$IG(S, A) := H(S) - \sum_{v \in Values(A)} \frac{|\{x \in S \mid value(x, A) = v\}|}{|S|} H(\{x \in S \mid value(x, A) = v\}) \tag{2.4}$$

where $Values(A)$ is the set of all possible values of the attribute A.

This can be rewritten as:

$$IG(S,A) = H(S) - \sum_{v \in Values(A)} p(A=v) \cdot H(S \mid A=v) \tag{2.5}$$

In information theory there is a quantity called mutual information that is used to measure the mutual dependence of two variables.

Definition 6. The **mutual information** between two variables X and Y, $I(X;Y)$ is given by

$$I(X;Y) := \sum_{x \in X} \sum_{y \in Y} p(x,y) \log \left(\frac{p(x,y)}{p(x)\,p(y)} \right) \tag{2.6}$$

It can be shown that

$$I(X;Y) = H(X) + H(Y) - H(X,Y) \tag{2.7}$$

Mutual information is also used in learning Bayesian Networks, section 4.2, and is the metric used by the LUMIN algorithm, see chapter 5, which demonstrates the interrelationships that can exist between different learners. Information gain is also related to the Kullback–Leibler divergence measure [Kullback & Leibler 1951].

Definition 7. The **Kullback-Leibler divergence** of two probability distributions P and Q of a discrete random variable, written as $D_{KL}(P \parallel Q)$ is defined as

$$\begin{aligned} D_{KL}(P \parallel Q) &:= \sum_i P(i) \log_2 \frac{P(i)}{Q(i)} \\ &= -\sum_i P(i) \log_2 Q(i) + \sum_i P(i) \log_2 P(i) \end{aligned} \tag{2.8}$$

where i steps through the values of the variable.

It can be shown that

$$D_{KL}(X \parallel Y) = IG(X,Y) \tag{2.9}$$

and

$$\begin{aligned} D_{KL}(P(X,Y) \parallel P(X)P(Y)) &= H(X) - H(X \mid Y) \\ &= H(X) + H(Y) - H(X,Y) \end{aligned} \tag{2.10}$$

So, Kullback–Leibler divergence is equivalent to information gain and both are related to mutual information. Information gain provides a useful metric, but this is only part of what is required

Algorithm 2.1 Summary of the ID3 Algorithm for Boolean Functions

1. ID3(Examples, Target_attribute, Attributes)
   Examples are the training examples. Target_attribute is the attribute whose value is to be predicted by the tree. Attributes is a list of other attributes that may be tested by the learnt decision tree. Returns a decision tree that correctly classifies the given Examples.
2. Create a Root node for the tree
3. If all Examples are positive, return a single-node tree Root, with label = +
4. If all Examples are negative, return a single-node tree Root, with label = -
5. If Attributes is empty, return the single-node tree Root, with label = most common value of Target_attribute in Examples
6. Otherwise Begin
   (a) A $\leftarrow$ the attribute from Attributes with the best information gain for the Target_attribute from Examples
   (b) The decision attribute for Root $\leftarrow$ A
   (c) For each possible value v_i of A
      i. Add a new tree branch below Root, corresponding to the test A= v_i
      ii. Let $Examples_{v_i}$ be the subset of Examples that have the value v_i for A
      iii. If $Examples_{v_i}$ is empty add a leaf node with label = most common value of Target_attribute in Examples
      iv. Else below this new branch add the subtree ID3($Examples_{v_i}$,Target_attribute,Attributes -{A})
7. End
8. Return Root

to build a decision tree from data.

The ID3 algorithm combines the information gain metric with a greedy search through the hypothesis space of all possible decision trees of increasing complexity. Information gain is used as a performance measure at each step to determine which attribute to choose next. The bias of ID3 is approximately that shorter trees are preferred to longer ones, and trees with high information gain near the root are preferred to those with low information gain near the root. An outline of ID3 is given in algorithm 2.1, and an example of how the ID3 algorithm builds trees is given in figure 2.3 using the data from table 2.1. The information gain measure can produce suboptimal trees when an attribute with a large number of values has little impact on the target attribute. Since the many-valued attribute will partition the examples into a large number of small sets, it can produce a significant information gain without actually helping with the classification. An alternative measure, gain ratio, was introduced by Quinlan [Quinlan 1986] to reduce this problem. This measure uses a term, split information, which is a measure of how evenly the attribute splits the examples with respect to the target attribute.

Definition 8. Split information is defined as:

$$SplitInformation(S,A) := -\sum_{i=1}^{c} \frac{|\{x \in S \mid value(x,A) = a_i\}|}{|S|} \log_2 \frac{|\{x \in S \mid value(x,A) = a_i\}|}{|S|} \tag{2.11}$$

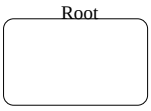
where a_i is the ith possible value of the c-valued attribute A. So, SplitInformation is actually the entropy of S with respect to the values of the attribute A.

Definition 9. Using SplitInformation we can define the **gain ratio** as follows:

$$GainRatio(S,A) := \frac{IG(S,A)}{SplitInformation(S,A)} \tag{2.12}$$

While the gain ratio is not a perfect selection method it does favour attributes which do not split the target attribute in a uniform manner. Further refinements to the use of gain ratio have been made and numerous alternative selection strategies have been tried with varying success [Lopez 1991, Mingers 1989a], and with adaptations to handle missing attributes [Quinlan 1993]. In addition work has been done on assigning different costs to the evaluation of attributes, either

Create a root node for the tree

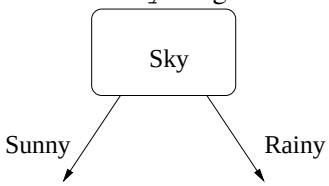


Not all examples of `EnjoySport` are positive
Not all examples of `EnjoySport` are negative
There are six attributes: `Sky, AirTemp, Humidity, Wind, Water`, and `Forecast`
The information gain for each attribute with respect to EnjoySport is:

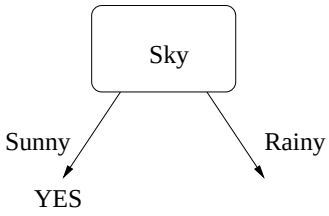
Attribute	Information Gain
Sky	0.811
AirTemp	0.811
Humidity	0.123
Wind	0.000
Water	0.123
Forecast	0.311

Since `Sky` and `AirTemp` both have the largest information gain we can choose either, we choose `Sky` to get:

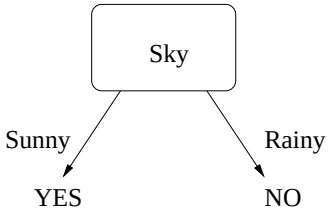
Adding branches for the possible values of `Sky` we get



For the `Sky` attribute with the value Sunny we have examples so we recurse on the procedure
Create a root node for our new (sub)tree
All examples of `EnjoySport(Sky=Sunny)` are positive so our new node becomes a leaf with the value `YES`



Okay we've finished with the `Sunny` branch and move on to the `Rainy` branch which also has values so again we recurse the procedure
Create a root node for our new (sub)tree
Not all examples of `EnjoySport(Sky=Rainy)` are positive
All examples of `EnjoySport(Sky=Rainy)` are negative so our new node becomes a leaf with the value `NO`



Having gone through all the values of the `Sky` attribute the tree is complete

Figure 2.3: Example of ID3 Algorithm on Data from Table 2.1

to mimic real world costs [Nunez 1991] or computational/time costs [Tan and Schlimmer 1990] & [Tan 1993].

One area in which significant improvements in the accuracy of decision tree learning can be made is that of post pruning. This is an attempt to reduce any overfitting which may have occurred during the learning process. One type of this is, reduced error pruning [Quinlan 1987], in which each node in the tree is considered for pruning. If a node is pruned then it is replaced with a leaf node whose classification is the most common classification of those training instances affiliated with the node. Nodes are only replaced if doing so does not degrade the performance of the tree with respect to the validation instances. This method is quite successful at improving the accuracy of the learnt tree over the whole of the instance space as it tends to remove nodes which were constructed due to particular regularities in the training data. Alternative tree pruning methods have also been explored as reported by [Mingers 1989b] and [Malerba et al 1995] including linear threshold risk bounds [Vapnik 1982, Anthony & Bartlett 1999], bounding the VC dimension[5] of the tree [Golea et al 1998], and bounding the subtrees [Kearns & Mansour 1998]. Pruning can also be considered in terms of the error bounds of the tree [Kääriäinen et al 2004, Zhong et al 2008]. There is also research into modifying the process of building a decision tree to take into account the error bounds of the tree [Mansour & McAllester 2000, Shah 2007], or additional constraints [Yin Wang & Wu 2004]. Another approach sometimes called, rule post pruning, is used by the C4.5 algorithm [Quinlan 1993]. Rule post pruning can be described as involving four steps

1. Infer the decision tree from the training set. No account of overfitting is taken at this point.
2. Convert the learnt tree into an equivalent set of rules. Each possible path from the root node to a leaf node is encoded as a separate rule.
3. Prune each rule by removing any preconditions that result in improving its estimated accuracy.
4. Sort the pruned rules by their estimated accuracy and consider them in this order when classifying new instances.

The above steps can be used in conjunction with disjoint training and validation instances. There are also heuristics which can be used when working with only training instances. Converting the

[5]This is a measure of the power/expressiveness of the learner [Vapnik & Chervonenkis 1971].

tree to rules is advantageous for people as it is usually easier to understand rules than a decision tree. It also removes the dependence of each node from its position in the tree. Each node becomes simply a part of one or more rules with no additional structure imposed.

The learning mechanism of decision trees can be modified to allow for learning in a multi-instance setting, one in which instances are not considered individually, but rather in groups called bags [Dietterich et al 1997]. Decision tree learners with various multi-instance learning modifications, bias modification [Blockeel and De Raedt 1998], with a looser definition of the multi-instance problem [Ruffo 2000], with a modified tree structure [Suzuki et al 2001], and with modified heuristics such as best-first node [Blockeel et al 2005], have been developed. Problem domains with only two classes can be dealt with by trees using a maximum margin classifier[6], this approach has also been developed for use with more than two classes [Tibshirani and Hastie 2007].

Decision trees are useful tools for predicting the outcome of new instances, but it is unclear that they provide much useful causal information. Interpreting causal information from the tree structure can be difficult. The usual criterion for node selection, such as the gain ratio, are at best only indirect measures of causality. The single parent nature of each node in the tree further obscures causal relationships as there is no tree structure which directly represents multiple causes for a variable. Figure 2.4 shows three different causal relationships between four variables and one decision tree. It is obvious that the decision tree is what would normally be expected from causal structures 2 and 3. In causal structure 1, if C has a much stronger influence on D than A or B, the contribution of A and B to a learnt tree could be pruned leaving the decision tree shown. This is a very simple example of the difficulties in trying to determine underlying causality from learnt decision trees.

Incorporating domain knowledge into decision tree learning is another area of difficulty. The basic greedy search strategy, examining each attribute independently, does not lend itself to the inclusion of domain knowledge. One method around this difficulty is to modify or expand the data to include the domain knowledge. This method was used by [Norton 1994] and [Salzberg 1995] with respect to examining DNA, and produced good results. It is also possible to modify the learning process to include explicit domain knowledge, this has been shown to be quite effective for a variation of the C4.5 algorithm in [Nazeri and Bloedorn 2004]. A slightly different approach demonstrated in [Karimi and Hamilton 2002, Console et al 2003] is the use

[6]Support Vector Machines are an example of maximum margin classifiers, see section 2.10.

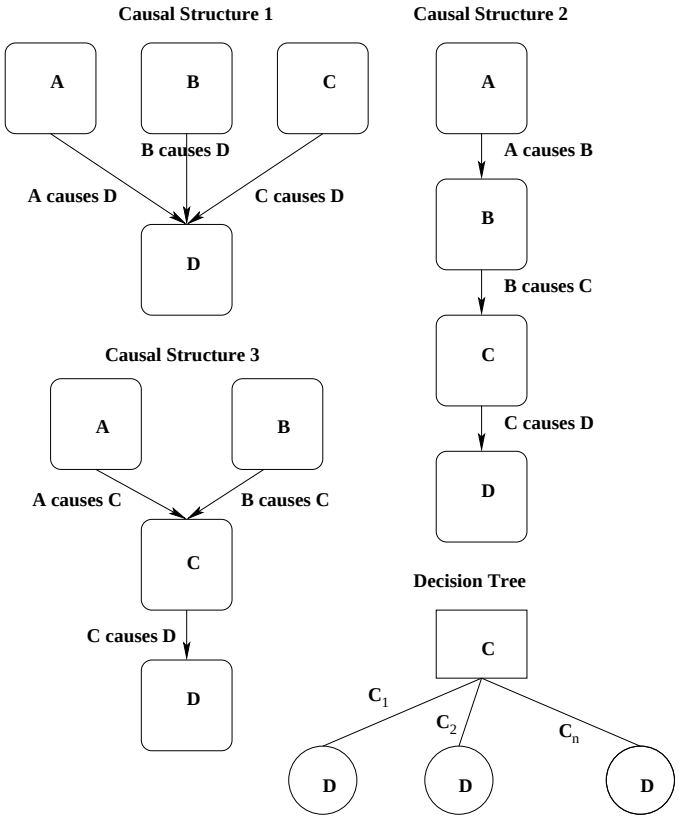


Figure 2.4: Three Causal Structures and a Possible Decision Tree

of temporal information not only in the structure of the data, but also to drive constraints on the trees that can be built.

2.4 K-Nearest Neighbour

In many ways K-Nearest Neighbour, k-NN, is the simplest form of learner. An object to be classified is compared to its k nearest neighbours and is given the same classification as the majority of its neighbours. k is a positive integer that is usually small. K-NN was introduced in an unpublished paper by Fix and Hodges [Fix and Hodges 1951]. K-NN uses some measure of distance, $d(x_1, x_2)$, between the data points x_1 and x_2. The distances between a test data item x_t and all the items in the training data set are computed. If the data items have a categorical classification then the classification of the test data item would be the most common classification of each of its k nearest neighbours, that is the mode classification, if a numeric classification was used the test point might be defined to have some average, possibly a weighted average, value of the value of the k nearest neighbours.

Let O be an ordered set of the n training data items $x_1,\ldots,x_n$ and x_t be the test data item such that $O=\{x_1,x_2,\ldots,x_n\}$ where $d(x_i,x_t)\leq d(x_{i+1},x_t)$. $f_c(x_i)$ is the classification of x_i then assuming the items have categorical classifications C where $C=\{c_1,c_2,\ldots,c_m\}$ then the classification of the test data item c_t would be:

$$c_t = \arg\max_{c\in C} \sum_{i=1}^{k} \begin{cases} 1 & \text{if } f_c(x_i)=c \\ 0 & \text{if } f_c(x_i)\neq c \end{cases} \tag{2.13}$$

where $\arg\max_{x\in X} f(x)$ is the value of x that maximises the value of $f(x)$[7]. It is common for K-NN to use the Euclidean distance, between the test item, x_t and the items in the training set, $x_1,\ldots,x_n$.

Definition 10. If the data items have p features then the **Euclidean distance** between x_t and x_i is defined as:

$$d(x_t,x_i) := \sqrt{(x_{t1}-x_{i1})^2+(x_{t2}-x_{i2})^2+\cdots+(x_{tp}-x_{ip})^2} \tag{2.14}$$

The distance metric is key to the effectiveness of the learner so there is research into the best measure of distance. It is likely that there is no single best metric and that the optimum distance measure is problem and possibly dataset related. Recent research has looked into creating distance metrics which cluster items of similar classification closely while separating differently classified items by large amounts. This approach is known as large margin nearest neighbour, (LMNN), classification was introduced by Weinberger and Blitzer [Weinberger et al 2006]. LMNN shares a number of properties with support vector machines, reducing the learning problem to convex optimisation, see section 2.10. Research into finding the best distance measure for a given dataset is an active topic [Xing et al 2003, Chopra et al 2005, Frome et al 2007], there is also some research into learning different metrics for different parts of the input space [Law and Zaniolo 2005, Weinberger & Saul 2008].

The Bayes risk, [Berger 1985], can be defined as follows:

Definition 11. For a variable X with a prior distribution π, let $\delta(x)$ be an estimator of X based on the data points $x\in X$. Let $C(X,\delta(x))$ be a cost function such as squared error.

[7]This ignores what to do in the event of a tie.

The *a-priori* **Bayes risk** of $\delta(x)$, R_δ, is defined as:

$$R_\delta := E_\pi \{C(X, \delta(x))\} \tag{2.15}$$

where the expectation is taken over the probability distribution of X. An estimator $\delta(x)$ is said to be a Bayes estimator, R_{Bayes}, if it minimises the Bayes risk among all estimators.

It can be shown, [Cover and Hart 1967], that for the 1-NN case where the size of the training set $n \to \infty$, with M categories, then the classification error R_{1NN} is bounded by the Bayes risk such that:

$$R_{Bayes} \leq R_{1NN} \leq R_{Bayes}\left(2 - \frac{M}{M-1} \cdot R_{Bayes}\right) \tag{2.16}$$

Nearest neighbour does not build a specific representation of the problem domain, instead it simply uses the training data and a similarity measure under the assumption that objects which are similar will be similarly classified. If the features of an object have radically different scales then features with a relatively small scale can become effectively insignificant. A transformation can be applied to the features to limit this distortion. Standardisation is a common such transformation which converts the original feature values into z-scores using the mean and standard deviation of the feature's values over all training examples, given by the relationship:

$$z_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j} \tag{2.17}$$

where x_{ij} is the value for the ith training example's jth feature, μ_j is the average of all x_{ij} for the jth feature, and σ_j is the standard deviation of all the x_{ij}. This transformation balances the weighting of the different features.

When k is greater than 1, such as 5-NN, the most common classification of the k nearest neighbours is used. A problem can occur here when it happens that some of the nearest neighbours are actually quite distant. It seems reasonable that close neighbours should have more weight than distant ones. Dudani [Dudani 1976], proposed just such a weighted K-NN learner. Like many learners K-NN suffers increasing computational cost with increasing dimensions. In order to reduce the computational cost of high dimensional data a number of data structures have been proposed. Friedman et al, [Freidman et al 1977], developed KD-trees which are a data structure where each k-dimensional non-leaf node has a splitting hyperplane that divides the data space into two subspaces. Points left of the hyperplane represent the left sub-tree of that node

and the points right of the hyperplane the right sub-tree. The hyperplane direction is chosen so that every node split to sub-trees is associated with one of the k-dimensions, and the hyperplane is perpendicular to that dimension vector. It is usual for the median value of the dataset in the dimension to be used for the splitting hyperplane creating a balanced KD-tree, but this may not be optimal in all cases and other values can be used when appropriate. KD-trees are usually searched with a simple branch and bound algorithm. It has been shown, [Lee and Wong 1977], that the worst case search time for an k-dimensional KD-tree containing M nodes is given by

$$t_{worst} = O\left(k.M^{1-\frac{1}{k}}\right) \tag{2.18}$$

If the dimensionality of the search space is high then searching KD-trees can be similar to an exhaustive search [Goodman et al 2004]. Other data structures have been developed to better cope with high dimensionality data such as the oct-tree, [Samet 1984], and metric ball tree [Omohundro 1989], and for Bregman Divergences, [Bregman 1967], a modified form of the ball tree called a Bregman Ball Tree has been shown to speed up searching [Cayton 2008].

Since basic K-NN does not construct a model of the problem domain it provides little useful information on any underlying causal relationships. It is possible to improve the performance of K-NN learners by a number of methods which use domain knowledge to restrict the available features, see section 2.13, modify the distance metric, or manipulate the input data space in a manner similar to that used in support vector machines.

2.5 Case-Based Learning

Case-based learning, often called case-based reasoning, CBR, has many similarities to K-Nearest Neighbour learning. They are both instance based learners, that is, they do not build some general model of the problem domain, but instead use specific examples as a guide to the likely solution for a new test case. Whereas K-NN learners use simple individual data points, often real values points in Euclidean space, for both the library of existing cases and the new test data item, case-based learners use rich symbolic descriptions for both their library instances and their test cases. Cased-based reasoning largely developed from the work of Roger Schank and his students in the 1980's, although there are numerous other influences such as analogical reasoning [Gentner 1983]. Initially CBR was looking at how people solve problems and was involved with the idea of dynamic memory [Schank 1982]. The basic idea is that people often approach a

problem by considering how they dealt with similar problems in the past. A separate group led by Bruce Porter were also developing case-based learners starting from more of an AI viewpoint looking at concept learning for classification tasks [Porter & Bareiss 1986]. In case-based reasoning we start with a database, or library, of existing problems and their possible solutions, both successful and unsuccessful. Case-based learners go through a number of steps when presented with a new case, these include:

Retrieve: Locating the case in the database closest to the new case. An issue here is that the rules for determining closeness tend to be very domain specific. There is some research which attempts to allow for more flexibility in this area, see for example [Montazemi & Gupta 1997].

Reuse: This involves taking a successful solution from the library case and adapting it to suit the current case. If the cases are very close no changes may be required otherwise some additional domain knowledge is required to either modify the previous solution for the current case, or to apply the method for generating a new solution.

Revise: The solution now needs to be tested, either directly or in a simulation. If necessary adjustments can be made, again this will require additional domain knowledge.

Retain: Lastly after a successful test of the solution, the new case and its solution can be added to the library of cases. This increases the domain information available for future cases.

The first demonstrated case-base system was Janet Kolodner's CYRUS [Kolodner 1983]. CYRUS could answer questions about the travels and meetings of former US Secretary of State Cyrus Vance. Case-based reasoning has proved useful in a number of real world applications including: CLAVIER [Hinkle & Toomey 1995], a system for determining efficient loads of composite material parts to be cured in an autoclave; SMART [Acorn & Walden 1992], a system to help the Compaq computer corporation deal with customer service calls; Vidur, a system for farmers in Manipur state in India, to help with paddy variety selection and weed control; and SQUAD, [Kitano et al 1992], Software Quality Control Advisor system which includes corporate structures and procedures in addition to case-based software.

An important issue for case-based learning is that of knowledge acquisition. There has to be some method of gaining the initial library of cases and solutions to be used with the new test cases. This is in many ways the same difficulty that all expert systems have with gaining initial knowledge. In addition case-based learners generate new case and solution pairs and this

knowledge also needs to be included in the library of cases so that it can be utilised for future test cases. It is not surprising then that there is research into knowledge acquisition related to case-based learning, [Sharma & Sleeman 1988, Bareiss 1989].

It is common for case-based learners to have knowledge in addition to that of just the case library. The learner will usually need to know how to adapt solutions from previous cases to cover a new test case. This kind of knowledge can be more like that found in a rule-based learner, see section 2.6, than the case knowledge. One area of research is to allow for more general knowledge rather than purely domain specific rules [Strube 1992].

As with most ML systems it is common to combine case-based learners with other learners to improve the performance of the overall system. The CASEY system combines a deep causal model with a case-based learner to diagnose heart disease. The case-based learner attempts to match new cases first, and if that fails the causal model is used to provide additional information. BOLERO [López & Plaza 1993] uses a rules-based system to store domain knowledge and a case-based planner to modify the search space used by the rules-based part. The INRECA system, [Manago et al 1993], uses both decision-trees and case-based reasoning. The case-based system acts as an interface and can help fill in missing data, like BOLERO the non case-based learner acts the the store of general knowledge, but unlike BOLERO the decision-tree generates an initial solution which the case-based system takes and then develops as necessary.

Case-based learners have no specific support for causal relationship discovery. However, it is possible to have causal rules as part of their domain knowledge. Since CBR systems expand their own library of cases based on the test cases they analyse, it would be possible for them to learn causal relationships. The use of causal rules, or causal models, to help with retrieval and adapting cases is not uncommon in CBR systems like CASEY, see also [Tighe & Tawfik 2008].

2.6 Rule-Based Learning

Rule sets are a very convenient method of representing information, and are generally easy for people to understand. It is possible to convert decision trees into rule form, and to learn rules using genetic algorithms. So, there are a number of indirect methods of generating sets of rules in addition to directly generating sets of rules.

A common strategy for learning a set of rules is to learn a rule which matches some of the required cases, remove the cases covered by the rule and then continue until there are no

unmatched cases left. This form of algorithm is called a sequential covering algorithm. Assume we have a function Learn-One-Rule that takes a set of training examples as input (positive and negative) and outputs a rule which covers a large number of the positive examples and maybe a few of the negative ones as well. Then we can remove the positive examples covered by the rule from our training set and invoke Learn-One-Rule again with the modified training set. This process can be repeated as required. In this manner we will get a disjunctive set of rules covering any desired proportion of the positive training examples. Once we have our final set of rules they can be sorted so that the most accurate rules are considered first. A typical example of a sequential covering algorithm is shown in algorithm 2.2. Then what is required is a suitable

Algorithm 2.2 A Typical Sequential Covering Algorithm

1. Sequential-Covering($Target\_attribute, Attributes, Examples, Threshold$)
2. $Learned\_rules \leftarrow \{\}$
3. $Rule \leftarrow$Learn-One-Rule($Target\_attribute, Attributes, Examples$)
4. while Performance($Rule, Examples$)$> Threshold$, do
   (a) $Learned\_rules \leftarrow Learned\_rules + Rule$
   (b) $Examples \leftarrow Examples - \{$examples correctly classified by $Rule\}$
   (c) $Rule \leftarrow$Learn-One-Rule($Target\_attribute, Attributes, Examples$)
5. $Learned_rules \leftarrow$sort $Learned\_rules$ according to Performance over $Examples$
6. Return $Learned\_rules$

Learn-One-Rule algorithm.

One approach to the Learn-One-Rule is a general to specific beam search. The algorithm shown in algorithm 2.3 is similar to that used by the program CN2 [Clark & Niblett 1989]. An example of Learn-One-Rule is shown in algorithm 2.4 using the data from table 2.1. An alternative might be to learn rules that only cover positive examples. Then any examples not covered by the rules would be assumed to be negative. This might be applicable in cases where a large majority of the cases were negative. However, in this case the performance measure would need to be changed from the entropy measure to something more like ratio of positive examples covered.

The AQ family of algorithms, [Michalski 1969] and [Michalski et al 1986], use another strategy. Unlike the sequential covering algorithm the AQ algorithms specifically learn rules to cover a specific target value. So, they will still end up with a set of disjunctive rules, but each rule will apply to a specific target value. Also the AQ algorithms use a specific positive example

Algorithm 2.3 General to Specific, Beam Search Version of Learn-One-Rule

1. Learn-One-Rule($Target\_attribute, Attributes, Examples, k$)
2. Initialise *Best_hypothesis* to the most general hypothesis $\emptyset$
3. Initialise *Candidate_hypotheses* to the set $\{Best\_hypothesis\}$
4. While *Candidate_hypotheses* is not empty, do
   (a) Generate the next more specific candidate_hypotheses
   (b) $All\_constraints \leftarrow$ the set of all constraints of the form $(a = v)$, where a is a member of *Attributes*, and v is a value of a that occurs in the current set of *Examples*
   (c) $New\_candidate\_hypotheses \leftarrow$
      i. for each h in *Candidate_hypotheses* do
         A. for each c in *All_constraints* do create a specialisation of h by adding the constraint c
   (d) Remove from *New_candidate_hypotheses* any hypotheses that are duplicated, inconsistent or not maximally specific
   (e) Update *Best_hypothesis*
   (f) for all h in *New_candidate_hypotheses* do
      i. if(Performance($h, Examples, Target\_attribute$)
      >Performance($Best\_hypothesis, Examples, Target\_attribute$)
      ii. then $Best\_hypothesis \leftarrow h$
   (g) Update *Candidate_hypotheses*
   (h) $Candidate\_hypotheses \leftarrow$ the k best members of *New_candidate_hypotheses*, according to the Performance measure
5. Return a rule of the form: IF *Best_hypothesis* THEN *prediction*, where *prediction* is the most frequent value of *Target_attribute* among those *Examples* that match *Best_hypothesis*

1. Performance($h, Examples, Target\_attribute$)
2. $h\_examples \leftarrow$ the subset of *Examples* that match h
3. Return –Entropy($h\_examples$), where entropy is with respect to *Target_attribute*

Algorithm 2.4 Example of Learn-One-Rule in Practise

Using data from table 2.1 with $Target\_attribute = EnjoySport$ and $k = 1$
Initialise $Best\_hypothesis$ to $\emptyset$
Initialise $Candidate\_hypotheses$ to the set $\{\emptyset\}$

Step 1

$All\_constraints = \{Sky = Sunny, Sky = Rainy, AirTemp = Warm, \ldots, Forecast = Change\}$
$New\_candidate\_hypotheses \leftarrow$
create new target hypotheses by adding each of the constraints to each of the current hypotheses
$h_1 = \{\emptyset\} \cup \{Sky = Sunny\}, h_2 = \{\emptyset\} \cup \{Sky = Rainy\}, \ldots, h_{11} = \{\emptyset\} \cup \{Forecast = Change\}$
removing hypotheses that are duplicated, inconsistent or not maximally specific gives
$New\_candidate\_hypotheses = \{Sky = Sunny, AirTemp = Warm\}$

Step 2

$Performance(Sky = Sunny, table\ 2.2, EnjoySport) > Performance(\emptyset, table\ 2.2, EnjoySport)$
$Best\_hypothesis = \{Sky = Sunny\}$
$Performance(AirTemp = Warm, table\ 2.2, EnjoySport) \not> Performance(Sky = Sunny, table\ 2.2, EnjoySport)$

Step 3

$Candidate\_hypotheses = \{Sky = Sunny, AirTemp = Warm\}$

Return Rule

IF Sky=Sunny THEN EnjoySport = YES

of the target value currently being investigated to guide the search. Only the attributes satisfied by the chosen positive example are considered during the search for specific hypotheses. It is common to call this type of algorithm separate-and-conquer the name used for it in papers by Giulia Pagallo and David Haussler [Pagallo & Haussler 1990] and Johannes Fürnkranz [Fürnkranz 1999]. The Learn-One-Rule algorithm performs a general to specific search, while algorithms like Find-S perform a specific to general search. There is only ever one most general case from which to start, but there may be many equally specific starting points. Selecting one of the multiple possible starting points risks increasing the probability of getting stuck at a local maximum. One approach to reduce the chance of this is to choose multiple starting points at random, and then use the best resulting hypothesis. This was approach was used by the GOLEM system [Muggleton & Feng 1990]. There is a lot of general interest in large margin classifiers partly due to the extensive theoretical examination such classifiers have undergone. Rückert and Kramer [Rückert and Kramer 2006] have developed a statistically motivated weighted rules learner which attempts to learn rules that maximise the margin between the training classes in a similar fashion to support vector machines, see section 2.10.

It is interesting to make some comparisons between algorithms like CN2 and ID3. While sequential covering algorithms learn one rule at a time, dealing with some subset of the training examples, algorithms like ID3 learn the entire set of disjuncts in one go, and all the examples are

used. We could therefore refer to algorithms like ID3 as simultaneous covering algorithms. ID3 partitions the data by attribute whereas CN2 uses subsets chosen using attribute-value pairs. To learn n rules each of which contain k attribute-value tests, CN2 will perform $n \cdot k$ independent search steps, but ID3 will perform fewer as each node in the tree can potentially be part of the preconditions for multiple rules.

Algorithms like CN2 suffer from the same problem of overfitting as decision tree algorithms like ID3. It is also possible to use a technique similar to post pruning to help with this. A non-training set of examples can be used as a pruning set. Then the rules generated by CN2 would have preconditions removed to increase their generality if this improved their performance on the pruning set of examples. This is analogous to post pruning in decision trees.

The Find-One-Rule algorithm is a generate and test algorithm, that is it generates its test cases based on what is syntactically correct in the hypothesis representation. Algorithms like Find-S are are data driven in that they use specific examples to revise the hypothesis. Data driven learning methods are more vulnerable to poor quality data than those using the generate and test approach.

Rule-based learning is an attempt to learn the domain rules governing the training examples and, hopefully, the problem domain as a whole. It is possible to add domain knowledge in a number of ways. As with all learners one application of knowledge is the choice of attributes. The construction of rules can also be tailored to fit with what is already known. One method of doing this used by the GRENDEL program [Cohen 1994], involves allowing the definition of the language to be used for defining the rules. In some ways this starts to overlap with explanation-based learning, section 2.11, as it is similar to explanation-based learning with a weak domain theory [Pazzani 1989].

Rule sets have the potential to provide direct explanations of the underlying mechanisms in the problem domain. Some research has focused on how best to learn rule sets that are understandable to humans [Dutch et al 2004]. However, the complexity of learnt rule sets can make comprehension of any underlying mechanisms almost impossible [Michalski 1983]. As with decision trees ensemble techniques, see section 2.17, can improve overall performance, but will generally lead to a loss of clarity [Cohen and Singer 1999]. It appears that while common, this complexity is not necessarily a feature of what should be learnt, learners which produce lightweight rule sets have been shown to be competitive [Weiss and Indurkhya 2000], and re-

search [Rückert & De Raedt 2008] has shown that simpler rule sets should be able to provide an accurate description of many problem domains. While the rules produced by the learners, hopefully, reflect true statements about the problem domain, there is no guarantee that they reflect the underlying causal structure of the domain. A poor selection of attributes could lead to rules which are a true reflection of the dataset, but are not related to the causal relationships in the problem domain. There has been some research into directly learning causal rules. [Pazzani 1993] gives an example of this using simple stories. However, in order to perform this causal learning, time information, in terms of a specified sequence of states, usually has to be supplied.

2.7 Inductive Logic Programming

Inductive Logic Programming, ILP, deals with first-order representations which allow variables in the expressions. This allows expressively powerful representations. An example of what is possible is

$$\begin{array}{llll} \text{IF} & Parent(X,Y) & \text{THEN} & Ancestor(X,Y) \\ \text{IF} & Parent(X,Z) \wedge Ancestor(Z,Y) & \text{THEN} & Ancestor(X,Y) \end{array}$$

where $Parent(X,Y)$ is true if Y is the parent of X, and $Ancestor(X,Y)$ is true if Y is the ancestor of X. This type of relationship is very hard to represent in propositional logic. If we tried to learn the second rule by giving a learning algorithm like ID3 or C4.5 a set of examples of parents and ancestors we would only end up with a set of specific rules applying to the examples we had provided, not the general rule given above.

PROLOG is a general purpose Turing-equivalent[8] programming language in which programs are expressed as collections of Horn clauses. In ILP it is usual to represent the rules in the form of Horn clauses. This is because in this form they can be used directly as programs in the PROLOG language. A clause is any disjunction of literals, where all variables are assumed to be universally quantified. A Horn clause is a clause containing at most one positive literal. An example of a Horn clause is

$$H \vee \neg L_1 \vee \neg L_2 \vee \ldots \vee \neg L_n$$

[8] That is having computational power equivalent to a universal Turing machine.

where H is a positive literal and $\neg L_1 \ldots \neg L_n$ are negative literals. Given the equalities

$$(B \vee \neg A) = (B \leftarrow A)$$

and

$$\neg(A \wedge B) = (\neg A \vee \neg B)$$

the Horn clause can be rewritten as

$$H \leftarrow (L_1 \wedge L_2 \wedge \ldots \wedge L_n)$$

which is equivalent to the rule

$$IF \quad L_1 \wedge L_2 \wedge \ldots \wedge L_n \quad THEN \quad H$$

The FOIL system [Quinlan 1990] uses the algorithm shown in algorithm 2.5. The FOIL system

Algorithm 2.5 The Basic FOIL Algorithm

1. FOIL($Target\_predicate, Predicates, Examples$)
2. $Pos \leftarrow$ those $Examples$ for which the $Target\_predicate$ is True
3. $Neg \leftarrow$ those $Examples$ for which the $Target\_predicate$ is False
4. $Learned\_rules \leftarrow \{\}$
5. while Pos do
   (a) Learn a $NewRule$
   (b) $NewRule \leftarrow$ the rule that predicts $Target\_predicate$ with no preconditions
   (c) $NewRuleNeg \leftarrow Neg$
   (d) while $NewRuleNeg$ do
      i. Add a new literal to specialise $NewRule$
      ii. $Candidate\_literals \leftarrow$ generate candidate new literals for $NewRule$, based on $Predicates$
      iii. $Best\_literal \leftarrow \arg\max_{L \in Candidate\_literals} Foil\_Gain(L, NewRule)$
      iv. add $Best\_literal$ to preconditions of $NewRule$
      v. $NewRuleNeg \leftarrow$ subset of $NewRuleNeg$ that satisfies $NewRule$ preconditions
   (e) $Learned\_rules \leftarrow Learned\_rules + NewRule$
   (f) $Pos \leftarrow Pos - \{$ members of Pos covered by $NewRule\}$
6. Return $Learned\_rules$

produces a set of Horn-like clauses. They differ from general Horn clauses in that the literals in the representation of its rules are not allowed to contain function symbols, and literals appearing in the body of the rule may be negated. The outer loop of the FOIL algorithm is very similar to the sequential covering algorithm, but the inner loop only builds rules which cover the cases where the target literal is true, the Learn-One-Rule algorithm also builds rules to cover cases where the target literal is false.

The process of creating generalisations for an existing rule within the FOIL system proceeds as follows. New literals are generated each of which may be added to the rule preconditions. If the current rule is

$$P(X_1, X_2, \ldots, X_k) \leftarrow L_1 \ldots L_n$$

where $L_1 \ldots L_n$ are the literals of the current precondition and where $P(X_1, X_2, \ldots, X_k)$ is the literal that forms the rule head. Then new literals L_{n+1} that have one of the following forms:

- $Q(V_1, \ldots, V_n)$, where Q is any predicate name occurring in *Predicates* and where the V_i are either new variables or variables present in the rule. At least one of the V_i in the created literal must already exist as a variable in the rule.
- $Equal(X_j, X_k)$, where X_j and X_k are variables already present in the rule.
- The negation of either of the above forms

To select the literal with the best apparent performance from the set of possible literals generated for each iteration, all possible bindings of each variable in the literal are considered. The rule is evaluated based on the sets of positive and negative variable bindings with preference given to rules that possess more positive bindings and fewer negative bindings. The evaluation used by FOIL to determine the success of adding any literal is based on the number of positive and negative bindings covered before and after adding the new literal. So, suppose we have a rule R and a candidate literal L. Let R' be the rule created by adding the literal L to rule R.

Definition 12. The value of **Foil_Gain**$(\mathbf{L}, \mathbf{R})$ of adding L to R is defined as:

$$Foil\_Gain(L,R) := t\left(\log_2 \frac{p_1}{p_1+n_1} - \log_2 \frac{p_0}{p_0+n_0}\right) \tag{2.19}$$

where p_0 is the number of positive bindings of rule R, n_0 is the number of negative bindings of rule R, p_1 is the number of positive bindings of rule R', and n_1 is the number of negative

bindings of rule R'. Lastly t is the number of positive bindings of rule R that are covered by rule R'.

When a new variable is introduced to R by L, then any original binding is considered to be covered so long as some binding extending it is present in the bindings of R'. From information theory we can conclude that $Foil\_Gain(L,R)$ is the reduction due to L in the number of bits needed to encode the classification of all positive bindings of R.

It is possible for FOIL to learn recursive rule sets such as that presented for $Ancestor(X,Y)$ earlier. To do this the target predicate, in this case $Ancestor(X,Y)$, needs to be added to the list $Predicates$. [Cameron-Jones & Quinlan 1993] show examples of where this has been successful and discuss issues with avoiding rule sets that produce infinite recursion.

To cope with noisy data FOIL uses a minimum description length approach to halt the growth of rules, in which new literals are added only when their description length is shorter than the description length of the data they explain. FOIL also uses post pruning of rules similar to that used for decision trees to help reduce overfitting. The details of its halting and pruning rules can be found in [Quinlan 1990].

Another approach to learning rule sets is to treat induction as inverted deduction [Jevons 1874]. This approach assumes that we have some inverse entailment operator $O(B,D)$ that takes the training data $D = \{\langle x_i, f(x_i)\rangle\}$ and background knowledge B as input and produces a hypothesis h as output such that

$$O(B,D) = h : (\forall \langle x_i, f(x_i)\rangle \in D)(B \wedge h \wedge x_i) \vDash f(x_i) \tag{2.20}$$

where $A \vDash B$ is read as A entails B. This definition subsumes the general case when no background knowledge is available and argues for learning methods that use the background knowledge to help guide the search. A downside is that, in general, formal logic systems are unable to deal with noisy data. Further first order logic is so expressive that the search through the hypothesis space is likely to be intractable and increasing background knowledge increases the complexity of the hypothesis space.

A general method for automated deduction is the resolution rule [Robinson 1965]. This is a sound and complete rule for deductive inference in first-order logic. It is possible to invert this rule to form an inverse entailment operator [Muggleton & Buntine 1988] which forms the core of the CIGOL program. We define a substitution to be any mapping of variables to terms.

For example the substitution $\theta = \{X/Bob, Y/Z\}$ indicates that the variable X is to be replaced by the term Bob, and that the variable Y is to be replaced by the term Z. We use the notation $W\theta$ to denote the result of applying the substitution θ to the expression W. For example if L is the literal $Father(X, Bill)$ and θ is the substitution defined above then $L\theta = Father(Bob, Bill)$. We call θ a unifying substitution for two literals L_1 and L_2, provided $L_1\theta = L_2\theta$. For example if $L_1 = Father(X, Y)$ and $L_2 = Father(Bill, Z)$, and $\theta = \{X/Bill, Z/Y\}$, then θ is a unifying substitution for L_1 and L_2 as $L_1\theta = L_2\theta = Father(Bill, Y)$. In first order resolution the resolvent of two clauses C_1 and C_2 is found by finding a literal L_1 from C_1 and a literal L_2 from C_2, such that some unifying substitution θ can be found for L_1 and $\neg L_2$. That is, $L_1\theta = \neg L_2\theta$. The resolution rule then constructs the resolvent C according to the equation

$$C = (C_1 - \{L_1\})\theta \cup (C_2 - \{L_2\})\theta \tag{2.21}$$

As an example let $C_1 = White(X) \leftarrow Swan(X)$ and $C_2 = Swan(Fred)$. First we rewrite C_1 in clause form as the expression $C_1 = White(X) \vee \neg Swan(X)$. Then we would find $L_1 = \neg Swan(X)$ from C_1, $L_2 = Swan(Fred)$ from C_2 and $\theta = \{X/Fred\}$. Thus $L_1\theta = \neg L_2\theta = \neg Swan(Fred)$ which demonstrates that θ is a unifying assumption. Therefore the conclusion C is the union of $(C_1 - \{L_1\})\theta$ and $(C_2 - \{L_2\})\theta$, that is $White(Fred) \cup \emptyset = White(Fred)$.

We can find the inverse by algebraic manipulation. First factor θ into θ_1 and θ_2, where $\theta = \theta_1\theta_2$, and θ_1 contains all substitutions involving variables from clause C_1, and θ_2 contains all substitutions involving variables from clause C_2. This factorisation is possible because C_1 and C_2 will always begin with distinct variable names as they are distinct universally quantified statements. Using the factorisation for θ we can restate the resolution equations as

$$C = (C_1 - \{L_1\})\theta_1 \cup (C_2 - \{L_2\})\theta_2 \tag{2.22}$$

If we restrict inverse resolution to infer only clauses C_2 that contain no literals in common with C_1 (i.e. a preference for the shortest C_2 clauses), then we can rewrite the equation as

$$C - (C_1 - \{L_1\})\theta_1 = (C_2 - \{L_2\})\theta_2 \tag{2.23}$$

then using the definition of the resolution rule that $L_2 = \neg L_1 \theta_1 \theta_2^{-1}$ we get the inverse resolution rule

$$C_2 = (C - (C_1 - \{L_1\}\theta_1)\theta_2^{-1} \cup \{\neg L_1 \theta_1 \theta_2^{-1}\} \tag{2.24}$$

This is a nondeterministic operator as we may find multiple choices for the clauses C_1 to be resolved and for the unifying substitutions θ_1 and θ_2. Each choice could potentially produce a different C_2.

Here is an example of the inverse resolution rule. Suppose we want to learn the predicate $GrandChild(Y,X)$, given the training data $D = GrandChild(Bob, Shannon)$ and the background information $B = \{Father(Shannon, Tom),\ Father(Tom, Bob)\}$. This is represented in figure 2.5. At the bottom of the diagram we set the conclusion C to the training example

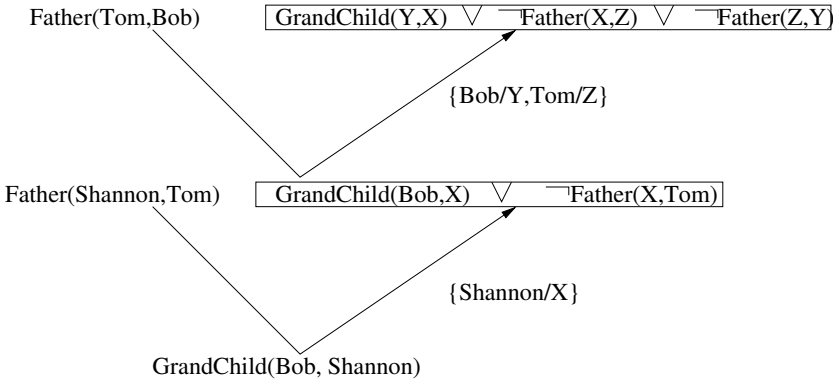


Figure 2.5: Example of Multistep Inverse Resolution

$GrandChild(Bob, Shannon)$ and choose $C_1 = Father(Shannon, Tom)$ from the background information. Since C_1 has only one literal we will have to use that. We choose $\theta_1^{-1} = \{\}$ and $\theta_2^{-1} = \{Shannon/X\}$. Then C_2 is the union of the clause $(C - (C_1 - \{L_1\})\theta_1)\theta_2^{-1} = C\theta_2^{-1} = GrandChild(Bob, X)$ and the clause $\{\neg L_1 \theta_1 \theta_2^{-1}\} = \neg Father(X, Tom)$. Thus the result is the clause $GrandChild(Bob,X) \vee \neg Father(X,Tom)$, which is equivalent to $GrandChild(Bob,X) \leftarrow Father(X,Tom)$. Similarly in the next step we use the background information $Father(Tom, Bob)$ and choose $\theta_1^{-1} = \{\}$ and $\theta_2^{-1} = \{Bob/Y, Tom/Z\}$ which leads to the required predicate

$$GrandChild(Y,X) \vee \neg Father(X,Z) \vee \neg Father(Z,Y)$$

and its equivalent

$$GrandChild(Y,X) \leftarrow Father(X,Z) \wedge Father(Z,Y).$$

This was, of course, choosing the better of many possible candidates for the literals and substitutions.

It is interesting to compare the inverse entailment and generate and test approaches. Inverse resolution will only generate hypotheses h that satisfy the relationship $(B \wedge h \wedge x_i) \models f(x_i)$ whereas generate and test, as used by FOIL, can generate any syntactically valid hypothesis. The generate and test algorithm has the advantage in being able to consider all the background knowledge when forming a hypothesis, inverse resolution can only consider a small amount of the information when generating its hypothesis at any one step. [Srinivasan et al 1995] provides one example of experimental research into comparing these two approaches.

ILP obviously makes extensive use of domain knowledge, without it an ILP learner could do little as it is the knowledge about the domain in the background data, combined with the training examples, that drives the learning process. Like most learners ILP performs best when only presented with relevant information [Qunilan & Cameron-Jones 1993]. If expert advise is available then ranking background information in a relevance ordering, relevant to the task at hand, can improve the performance of an ILP learner [Srinivasan, King & Bain 2003]. In its pure form ILP excludes probabilistic and statistical reasoning. However, combining it with kernels opens the opportunity to perform statistical reasoning [Passerini et al 2006] and this can be extended to a wider variety of learning techniques [Sato 1995, Getoor & Taskar 2007, Sato et al 2008]. In addition the field of Markov Logic Networks [Richardson & Domingos 2006], MLN, combines first order logic with probabilistic graphical models. Research into probabilistic variations of ILP is an active field [De Raedt et al 2008, Chen et al 2008].

There is no direct support in ILP for causal rules. While ILP learners can learn causal relationships they cannot distinguish causal and non-causal relationships. In the above example the rule $GrandChild(Y,X) \leftarrow Father(X,Z) \wedge Father(Z,Y)$ could be considered to be either causal or declarative. The rules discovered by ILP are easily understood so its provides a good foundation to extend existing knowledge in a given domain with the possibility of discovering causal relationships.

2.8 Artificial Neural Networks

Artificial Neural Networks, ANNs, are biologically inspired learning systems. ANNs are used both as a tool to learn specific target functions and as a means of modelling and understanding

natural neural networks[9]. In many complex real-world tasks ANNs are among the most effective learning methods currently known. An ANN consists of a number of units arranged in layers. Each unit can have a number of inputs and a single output. In general the system inputs go into the first layer, and the output of each unit in a layer acts as an input for all the units in the following layer. Since the output of all but the last layer are not directly available, these layers are called hidden layers. An examples of a ANN is shown in figure 2.6. This is taken from

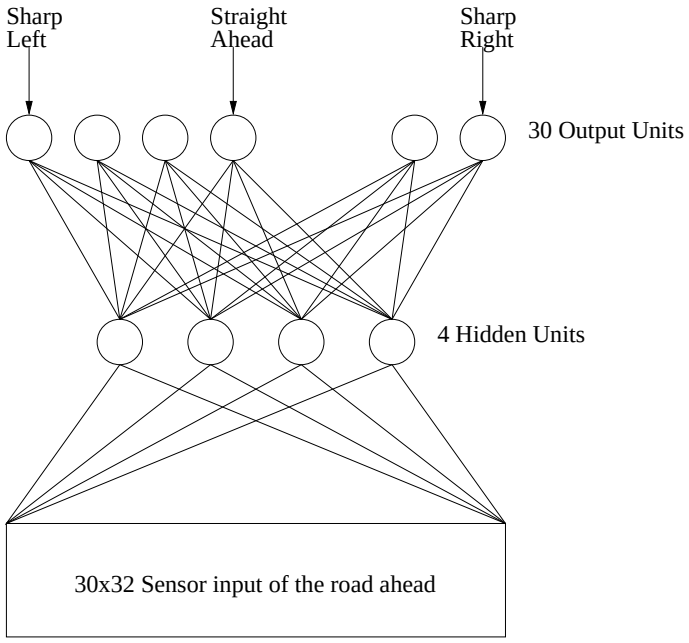


Figure 2.6: Neural Network Examples Taken from the ALVINN System

[Pomerleau 1993] and relates to the ALVINN system which uses a ANN to steer an autonomous vehicle on the public highway.

The name of a unit in the network is determined by the function which relates its inputs to its output. Probably the three most widely used units are the perceptron [Rosenblatt 1957], the linear and sigmoid threshold units. The perceptron shown in figure 2.7 is the type of unit most closely associated with biological neurons. A perceptron takes a vector of real-valued inputs calculates a linear sum and outputs a 1 if the value exceeds some threshold and -1 otherwise. If we define the inputs to be X_1 to X_n, the weights to be w_0 to w_n then the output $o(X_1,...,X_n)$ would be calculated as follows

[9]For more information on their use in biological modelling see [Gabriel and Moore 1990] and [Churchland and Sejnowski 1992].

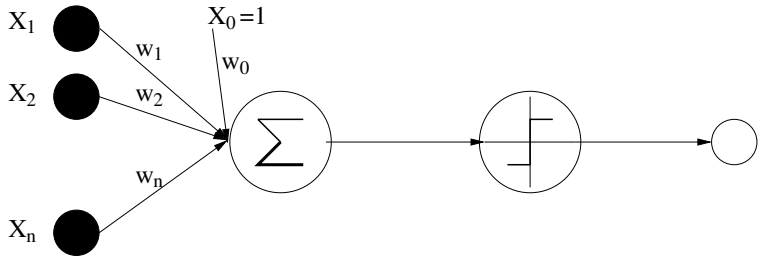


Figure 2.7: A Perceptron

$$o(X_1,...,X_n) = \begin{cases} 1 \text{ if } w_0 + w_1X_1 + w_2X_2 + ... + w_nX_x > 0 \\ -1 \text{ otherwise} \end{cases} \tag{2.25}$$

A single perceptron can represent the boolean functions AND, OR, NAND, and NOR. As any boolean function can be represented by some combination of these functions, a two layer perceptron network can represent any boolean function.

A linear unit is simply a perceptron where rather than comparing the sum with a threshold to determine the output value, the sum itself is used as the output value. That is, for a linear unit

$$o(X_1,...,X_n) = w_0 + \sum_{i=1}^{n} w_iX_i \tag{2.26}$$

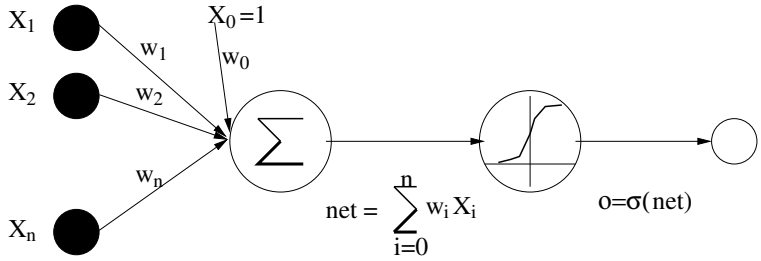


Figure 2.8: A Sigmoid Threshold Unit

A sigmoid unit shown in figure 2.8 is like a perceptron in that it computes the sum of its input, but it then computes its output as a continuous differentiable function of this sum. The output is computed as follows:

$$o(\bar{w},\bar{X}) = \sigma(\bar{w}\cdot\bar{X}) \tag{2.27}$$

where

$$\sigma(Y) = \frac{1}{1+e^{-Y}} \tag{2.28}$$

Training a perceptron can be done by updating the input weights to make the output more closely match the required output. The perceptron training rule can be represented as

$$w_i \leftarrow w_i + \Delta w_i \tag{2.29}$$

where

$$\Delta w_i = \eta \left(t - o\left(w_i, X_i\right)\right) X_i \tag{2.30}$$

t is the target output value for the current training example, $o\left(w_i, X_i\right)$ is the output value produced by the current weights for the current example, η is a positive constant called the learning rate. The learning rate determines the rate at which the weights change. It is fairly clear that this rule will adjust the weights in a way which will change the output of the perceptron to be more in line with the training examples. However, for linearly separable training examples and with a sufficiently small value of η it has been proven to converge within a finite number of steps [Minsky & Papert 1969]. The delta training rule is a more sophisticated rule which allows convergence towards a solution with non-linearly separable training examples. The delta rule uses the idea of gradient descent, and training error.

Definition 13. For a linear unit we could define the **training error**, E, to be:

$$E\left(\bar{w}\right) := \frac{1}{2} \sum_{d \in D} \left(t\left(d\right) - o\left(\bar{w}, d\right)\right)^2 \tag{2.31}$$

where D is the set of training examples, $t\left(d\right)$ is the target value for training example d, and $o\left(\bar{w}, d\right)$ is the actual output for example d with the weights $\bar{w}$.

Definition 14. The gradient of E with respect to $\bar{w}$ written $\nabla \mathbf{E}\left(\bar{\mathbf{w}}\right)$ is defined as

$$\nabla E(\bar{w}) := \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \cdots, \frac{\partial E}{\partial w_n}\right] \tag{2.32}$$

Then the gradient descent rule is

$$\bar{w} \leftarrow \bar{w} + \Delta \bar{w}$$

where

$$\Delta \bar{w} = -\eta \nabla E$$

So on an individual weight basis we get

$$\Delta w_i = \eta \sum_{d \in D} (t_d - o(w_i, d)) X_{id} \tag{2.33}$$

where X_{id} denotes the single input component x_i for training example d. We can then define an algorithm to perform gradient descent as shown in algorithm 2.6. Each training example is a pair $\langle \bar{X}, t \rangle$ where $\bar{X}$ is a vector of input values and t is the target output value.

Algorithm 2.6 Gradient Descent Algorithm for Training a Linear Unit

1. Gradient_Descent(training_examples,η)
2. Initialise each w_i to some small random value
3. Until the termination condition is met. Do
   (a) Initialise each Δw to zero
   (b) For each $\langle \bar{X}, t \rangle$ in training_examples. Do
      i. Input the instance $\bar{x}$ to the unit and compute the output o
      ii. For each linear unit weight w_i, Do
   (c) $$\Delta w_i \leftarrow \Delta w_i + \eta (t - o(w_i, X_i)) X_i$$
   (d) For each linear unit weight w_i, Do
   (e) $$w_i \leftarrow w_i + \Delta w_i$$

The gradient descent training rule suffers from some general gradient descent problems. If there are local minima it may get stuck there and miss the global minimum. If the step size is too large it can miss the minimum, if its too small it may not converge to a minimum in a reasonable number of steps. This gradient descent algorithm only updates the weights once for each pass through all the training examples making it quite computationally expensive for large training sets. One alternative approach is to use a stochastic approximation to the gradient descent rule. An example of this is to define an error function, $E_d(\bar{w})$, for each training example d.

Definition 15. The **stochastic training error** function, E_d, could be defined as

$$E_d(\bar{w}) := \frac{1}{2}(t_d - o(\bar{w}, d))^2 \tag{2.34}$$

where t_d is the target value for the training example d, and $o(\bar{w}, d)$ is the output value for that example.

This leads to a very similar algorithm, except that the weights in the vector $\bar{w}$ are updated after each training example, and the new weights are immediately used for the next example. The stochastic approximation is less computationally expensive and, because it updates for each example in turn thus moving away a little from the actual error surface defined by the problem, is somewhat less likely to fall into local minima. However it is not using the true gradient and so may not find the minimum, and should probably be used with a smaller step size than would be used for the gradient descent algorithm.

The training algorithm shown above is only defined for single units. It is more common to have multi-layer networks due to their representational power. Every bounded continuous function can be approximated to arbitrary accuracy by a network with two layers, a hidden layer of sigmoid units and an output layer of linear units [Cybenko 1989] & [Hornick et al 1989]. Any function can be approximated to arbitrary accuracy by a three layer network with two hidden sigmoid layers and a linear output layer [Cybenko 1988]. It is possible to construct a gradient descent rule for a multilayer network in the same fashion as for a single unit. The error term for a multiple unit network, $E(\bar{w})$, has to take into account the errors of all the outputs.

Definition 16. We define the error term for a multiple unit network, $\mathbf{E}(\bar{\mathbf{w}})$, as

$$E(\bar{w}) := \frac{1}{2}\sum_{d \in D}\sum_{k \in outputs}(t_{kd} - o_{kd})^2 \tag{2.35}$$

where outputs is the set of output units in the network, and t_{kd} and o_{kd} are the target and output values associated with the kth output unit and training example d.

Then the stochastic version of the gradient descent algorithm, usually called the BACKPROPAGATION algorithm, [Rumelhart et al 1986], in connection with feed-forward neural networks, is given by algorithm 2.7. Each training example is a pair of the form $\langle \bar{X}, \bar{t} \rangle$, where $\bar{X}$ is the vector of input values and $\bar{t}$ is the vector of target network output values, η is the learning rate, n_{in} is the number of network input units, n_{out} is the number of network output units, $\bar{n}_{hidden}$ is

a vector of the number of hidden units so that $\bar{n}_{hidden_l}$ is the number of hidden units in hidden layer l, and L is the number of hidden layers. The BACKPROPAGATION algorithm can be adapted to work for any acyclic feed-forward network. It is difficult to accurately characterise

Algorithm 2.7 The Stochastic Version of the BACKPROPAGATION Algorithm

1. BACKPROPAGATION(training_examples,η,n_{in},n_{out},$\bar{n}_{hidden}$,L)
2. Create a feed-forward network with n_{out} output units, $\bar{n}_{hidden_l}$ hidden units in hidden layer l, so that $1 \leq l \leq L$, and n_{in} input units. The input from unit i into unit j is denoted x_{ij}, and the weight from unit i to unit j is denoted w_{ji}
3. Initialise all the network weights to small random values
4. Until the termination condition is met. Do
   (a) For each $\langle \bar{X}, \bar{t} \rangle$ do
      i. Input the instance $\bar{X}$ to the network and propagate through the network computing the output o_u of every unit u in the network.
      ii. For each network output unit k, calculate its error term δ_k
      iii.
      $$\delta_k \leftarrow o_k(1-o_k)(t_k-o_k)$$
      iv. For each layer of hidden units l
         A. For each hidden unit in the layer h calculate its error term δ_h
         B.
         $$\delta_h \leftarrow o_h(1-o_h)\sum_{s \in layer\, l+1} w_{sh}\delta_s$$
         C. Update the network weights w_{ij}
         D.
         $$w_{ij} \leftarrow w_{ij} + \Delta w_{ij}$$
         where
         E.
         $$\Delta w_{ij} = \eta \delta_j x_{ji}$$

the bias of the BACKPROPAGATION algorithm, but it can be said to approximate to smooth interpolation between points. BACKPROPAGATION can be a slow learner, difficult to optimally configure, [LeCun et al 1998] examines methods to optimise its performance. BACKPROPAGATION is not the only network training rule, both amended versions of BACKPROPAGATION and other learning rules have been developed [Simard et al 1992, Mitchell & Thrun 1993, Riedmiller & Braun 1993, Igel & Husken 2003, Wojnarski 2007]. Changing the target from a simple binary to ranked options [Caruana et al 1996] can improve accuracy, as can directly learning the classification rather than reducing the error function [Rimer and Martinez 2006]. Ordinal data are often treated simply as discrete classes, but maintaining their ordering can aid

in the learning process [da Costa & Cardoso 2005]. Sensitivity analysis has been used both to help with initial weights and in developing alternative learning algorithms [Castillo et al 2006]. Also variations in the error rule and adaption of learning parameters have been investigated, see [Bishop 1996]. Deep neural networks, those with many hidden layers, are useful in representing highly-varying target functions. However, the usual training problems, such as falling into local minima, are often magnified when training such a network. There is some research into methods to effectively train deep networks [Larochelle et al 2009]. Deep Belief Networks, DBNs, are an alternative form of neural network which have gained some popularity, particularly for working with images [Ranzato et al 2007, Bengio et al 2007] and video [Sutskever & Hinton 2007, Taylor et al 2007]. DBNs are probabilistic generative models that are composed of multiple layers of stochastic, latent variables. The latent variables typically have binary values and are often called hidden units or feature detectors. DBNs are usually trained a layer at a time and pose their own difficulties from a learning perspective. There is active research into learning algorithms for DBNs [Hinton et al 2006, LeCun & Bengio 2007]. The learning rules presented so far result in a single network. An alternative type of learner called a Bayesian Neural Network, BNN, produces what is in effect a weighted average over all possible weights, that is, all networks. As with all Bayesian methods a prior value, or more correctly a prior probability distribution, is required for the item(s) under investigation, in this instance the network weights. An issue with BNNs is how to select these priors. Hierarchical priors often reflecting meta-features have been used with some success [MacKay 1992, Neal 1996, Andrieu et al 2001], and in the absence of additional knowledge flat priors can also produce good results [Lee 2003].

One of the interesting features of ANNs is that after training the hidden units often act as feature selectors. That is, the output of specific hidden units can act as an indication of the presence of a given feature of the problem. While this is potentially useful it also exposes one of the weaknesses of ANNs, it is difficult to interpret the meaning of the network weights. So, while it is possible that during training the ANN has identified important elements of the problem, it is not easy to express that information in an understandable form.

The previous discussion only mentioned feed-forward, acyclic static neural networks. There has been research into recurrent networks, that is, networks which have a feedback loop. Information on training methods can be found in [Jorden 1986, Elman 1990, Williams & Zipser 1995]. Detail on amending the BACKPROPAGATION algorithm to train recurrent networks can be

found in [Mozer 1995]. Recurrent networks are of particular interest in modelling time dependent systems and various types of Temporal-Difference, TD, networks have been researched to aid with this task [Dayan 1993, Sutton & Tanner 2004, Makino 2009]. Another area of interest is in dynamically altering the structure of the network. [Fahlam & Lebiere 1990] show how the CASCADE-CORRELATION algorithm grows a hidden layer of units as part of the learning process. An alternative approach, called Optimal brain damage, in which the network is altered by the removal of network links is given in [LeCun et al 1990].

The previous ANN learners take no account of any existing domain knowledge, the following learners combine domain knowledge with ANN learners. Its is possible to use prior knowledge in a number of ways, if there is information on the form of the target function this can be used to constrain the learner [Dugas et al 2009]. Another approach is to use the domain theory to construct an initial hypotheses, then use an inductive learning method to modify this initial hypothesis to better fit the data. This is the approach of the KBANN algorithm [Shavlik & Towell 1989] and the Tyling-Pyramid algorithm [Parekh and Honavar 1998]. In KBANN an ANN is constructed that will always classify instances according to the domain theory. This ANN is then trained on examples using BACKPROPAGATION. KBANN assumes the domain theory is in the form of a set of nonrecursive, propositional Horn clauses. KBANN works in two stages, in the first an ANN is created and its weights assigned so that its output will perfectly match the domain theory for any given test case, in the second stage the ANN is trained using the BACKPROPAGATION algorithm. The KBANN algorithm is shown in algorithm 2.8.

KBANN allows for improved generalisation over BACKPROPAGATION particularly when the domain knowledge is approximately correct and training data is scarce. KBANN has been shown to outperform purely inductive systems for several real world problems [Towell et al 1990]. In addition it has been shown, [Fu 1993, Towell et al 1990], that Horn clauses extracted from the final trained network provide a refined domain theory that better fits the observed data. It is not in general possible to extract Horn clauses from a neural network. [Craven & Shavlik 1994] and [Craven 1996] describe alternative methods for extracting symbolic rules from trained networks. KBANN can perform worse than purely inductive systems if the domain knowledge is highly inaccurate and there are few training examples. KBANN and similar algorithms are still being researched and developed [Jagadeesh et al 2008] and its updated versions remain competitive.

Algorithm 2.8 The KBANN Algorithm

1. KBANN(*DomainTheory*, *TrainingExamples*)
2. `Analytical Step: Create an initial network equivalent to the domain theory`
3. For each instance attribute create a network input
4. For each Horn clause in the *DomainTheory* create a network unit as follows
   (a) Connect the inputs of this unit to the clause antecedents
   (b) For each non-negated antecedent of the clause, assign a weight of W to the corresponding sigmoid unit input
   (c) For each negated antecedent of the clause, assign a weight of $-W$ to the corresponding sigmoid unit input
   (d) Set the threshold weight w_0 for this unit to $-(n-0.5)W$, where n is the number of non-negated antecedents of the clause
5. Add additional connections among the network units, connecting each network unit at a depth i from the input layer to all network units at a depth $i+1$. Assign random near-zero weights to these additional connections.
6. `Inductive Step: Refine the initial network`
7. Apply the BACKPROPAGATION algorithm to adjust the initial network weights to fit the *TrainingExamples*.

An alternative way of using domain knowledge is to include it in the error function to be minimised by gradient descent. One technique that has been investigated is using prior knowledge in the form of derivatives of the target function. An example of this is that TANGENTPROP algorithm. This trains a neural network to fit both the training values and the training derivatives. In most cases we consider training data to be of the form $\langle x_i, f(x_i)\rangle$ where x_i is an instance and $f(x_i)$ is its training target value. The TANGENTPROP algorithm assumes that training derivatives of the target function are also provided. So, if each instance x_i is described by a single real value, then each training example may be of the form

$$\left\langle x_i, f(x_i), \left.\frac{\partial f(X)}{\partial X}\right|_{x_i} \right\rangle \tag{2.36}$$

where $\left.\frac{\partial f(X)}{\partial X}\right|_{x_i}$ denotes the derivative of the function f with respect to X, evaluated at the point $X = x_i$.

Consider the task of learning to recognise handwritten characters. We will assume that the input x corresponds to an image containing a single character, and the task is to classify the character. We determine that the character is invariant to small rotations of the character within the image. We define a transformation $s(\alpha, X)$ that rotates the image X by α degrees in a clockwise direction. We can now assert rotational invariance by stating that for each training instance x_i the derivative of the target function with respect to this transformation is zero, that is for each training example x_i

$$\frac{\partial f(s(\alpha, x_i))}{\partial \alpha} = 0 \tag{2.37}$$

where f is the target function and $s(\alpha, x_i)$ is the image resulting from applying the transformation s to the image x_i. At this point its useful to recall the error function used by gradient descent in BACKPROPAGATION to minimise the sum of the squared errors

$$E = \sum_i \left(f(x_i) - \widehat{f}(x_i) \right)^2 \tag{2.38}$$

where x_i denotes the ith training instance, f denotes the true target function, and $\widehat{f}$ denotes the function represented by the learnt neural network. We might wish to have any number of transformations to include as additional information, for example to denote translational invariance. Each transformation must be of the form $s_j(\alpha, X)$ where α is a continuous parameter, where s_j is

differentiable, and where $s_j(0,X) = X$. For each such transformation, $s_j(\alpha,X)$, TANGENTPROP considers the squared error between the specified training derivative and the actual derivative of the learnt network. The modified error function is

$$E = \sum_i \left[\left(f(x_i) - \widehat{f}(x_i) \right)^2 + \mu \sum_j \left(\frac{\partial f(s_j(\alpha,x_i))}{\partial \alpha} - \frac{\partial \widehat{f}(s_j(\alpha,x_i))}{\partial \alpha} \right)^2_{\alpha=0} \right] \tag{2.39}$$

where μ is a constant provided by the user to determine the relative importance of fitting the training values versus fitting training derivatives. The gradient descent rule for minimising this error function can be found in [Simard et al 1992]. The TANGENTPROP algorithm is not resilient to errors in the training derivatives. Depending on the value of μ it can perform worse than BACKPROPAGATION when given incorrect derivatives.

The EBNN (Explanation-Based Neural Network learning) algorithm [Mitchell & Thrun 1993] and [Thrun 1996] is an extension of the TANGENTPROP algorithm in two significant ways: instead of the user computing the training derivatives, EBNN computes the derivatives itself for every training example; EBNN automatically assigns a value for μ the relative importance of the inductive and analytical components of learning, and does so individually for each training example. The derivatives are calculated by explaining each training example in terms of a given domain theory, then extracting the derivatives from this explanation. The value of μ is based on a heuristic that considers how well the training example is explained by the domain theory. When the example accurately reflects the domain theory the derivative is given a high weight, when the domain theory poorly explains the training example the derivative is given a low weight. The input to EBNN are the training examples, $\langle x_i, f(x_i) \rangle$, with no derivatives, and the domain theory, analogous to that used in KBANN, but in the form of trained neural networks rather than Horn clauses. The output is a trained neural network, trained to match the target function, f, based on the examples and the domain theory derived derivatives. EBNN invokes KBANN to train the neural network, but supplies its calculated value for the derivative and μ. To calculate a derivative for each training example, $\langle x_i, f(x_i) \rangle$, EBNN calculates a partial derivative for the predicted value for each attribute of the example. This set of derivatives is the gradient of the domain theory prediction function with respect to the input instance. This matrix of gradients is called the Jacobian of the target function. An outline of the EBNN algorithm is shown in algorithm 2.9. EBNN appears to be more robust than TANGENTPROP to errors in the computed derivatives [Masuoka 1993].

Algorithm 2.9 The EBNN Algorithm

1. EBNN(*DomainTheory*, *TrainingExamples*)
2. Create a fully connected feedforward neural network with the same structure as that constructed by KBANN, but with all the weights initialised to small random values
3. for each training example $\langle x_i, f(x_i)\rangle$ do
   (a) Using the domain theory calculate its predicted value of the target function $A(x_i)$
   (b) Analyse the weights and activations of the domain theory networks to extract the derivatives of $A(x_i)$ with respect to each of the attributes of x_i, i.e. the Jacobian of $A(X)$ evaluated at $X = x_i$.
   (c) Train the target network to fit the error function

$$e = \sum_i \left[\left(f(x_i) - \widehat{f}(x_i) \right)^2 = \mu_i \sum_j \left(\frac{\partial A(x)}{\partial x^j} - \frac{\partial \widehat{f}(x)}{\partial x^j} \right)^2_{(x=x_i)} \right]$$

   where

$$\mu_i \equiv 1 - \frac{|A(x_i) - f(x_i)|}{c}$$

   where x_i denotes the ith training instance, $A(x)$ denotes the domain theory prediction for input x, the superscript notation x^j denotes the jth attribute of the vector x and the coefficient c is a normalising constant chosen such that $\forall i, 0 \leq \mu_i \leq 1$

Another way of using domain knowledge is to use it to alter the hypothesis space search, by defining legal operators with which to search the hypothesis space. An example of this type of approach is the FOCL system [Pazzani et al 1991] and [Pazzani & Kibler 1992]. FOCL has a lot of similarity to the FOIL system, see section 2.7. Both learn a set of first-order rules to cover the training examples. Both systems employ a sequential covering algorithm that learns a single Horn-like clause, removes all positive training examples covered by the new clause and then iterates the procedure over the remaining examples. Both systems create each new rule by following a general-to-specific search, starting with the most general Horn clause, i.e. one with no preconditions. A number of candidate specialisations of the candidate clause are then generated and the specialisation with the greatest information gain relative to the training examples is selected. This process is repeated generating further candidate specialisations until a Horn-like clause with satisfactory performance is obtained. The difference lies in the method of generation of the candidate specialisations. FOIL generates new candidate specialisations by adding a single new literal to the clause preconditions. FOCL uses the same technique and in addition generates extra specialisations based on the domain theory. Literals can be classified in to two classes. An operational literal is one that can be used as part of an output hypothesis,

that is, one that only refers to the attributes in the training examples. Literals that occur only as intermediate features in the domain theory are non-operational. At each point in its general-to-specific search, FOCL expands its current hypothesis h using the following two operators:

1. For each operational literal that is not part of h, create a specialisation of h by adding this single literal to the preconditions. This is the same as the standard FOIL operator.

2. Create an operational, logically sufficient condition for the target concept according to the domain theory. Add this set of literals to the current preconditions of h. Prune the preconditions of h by removing any literals that are unnecessary according to the training data.

The detailed operation of the second operator is as follows. FOCL first selects one of the domain theory clauses whose head (postcondition) matches the target concept. If there are several such clauses it selects the clause whose body (preconditions) has the highest information gain relative to the training examples of the target concept. The preconditions of the selected clause form a logically sufficient condition for the target concept. Each non-operational literal in this sufficient condition is now replaced, again using the domain theory and substituting clause preconditions for clause postconditions. This process continues until the sufficient conditions have been related in terms of operational literals. If there are alternative domain theory clauses that produce different results, then the one with the greatest information gain is greedily selected at each step of the process. As a final step this sufficient condition is pruned. For each literal in the expression, the literal is removed unless its removal reduces classification accuracy over the training examples. This step helps to avoid over-specialisation in the case of imperfect domain theory. The remaining set of literals is added to the precondition of the current hypothesis. Once the candidate specialisations have been generated using both rules, the candidate with the best information gain is selected. The search then continues by considering further specialisations of the theory-suggested preconditions, thereby allowing the inductive component of learning to refine the preconditions derived from the domain theory. So FOCL learns Horn clauses of the form

$$c \leftarrow o_i \wedge o_b \wedge o_f$$

where c is the target concept, o_i is an initial conjunction of operational literals added one at a time by the first syntactic operator, o_b is a conjunction of operational literals added in a single

step based on the domain theory, and o_f is a final conjunction of operational literals added one at a time by the first syntactic operator. Any of these three sets of literals may be empty.

In their initial incarnations ANNs were opaque, if efficient, learners which took little notice of knowledge in the problem domain. However, there are now many systems in which ANNs make extensive use of domain knowledge both in building and training the network. It is possible to extract various information from a trained network for example symbolic rules [Thrun 1993, McMillan et al 1992, Andrews et al 1995, Dutch et al 2001, Castro et al 2002, Odajima et al 2008] or time series models [Zyl and Omlin 2001]. ANNs in an ensemble, see section 2.17, pose a different problem regarding rule extraction and there is research in that area [Wall & Cunningham 2000]. An ANN can be constructed without data using available knowledge [McGarry & Wermter 2005]. ANNs can even be used to help improve domain knowledge by building a network from domain knowledge, training it and then re-examining the knowledge in the network(s) [Zhou et al 2003, Parekh and Honavar 1998].

2.9 Bayesian Learning

Bayesian learning is based on the assumption that the quantities of interest are governed by probability distributions. Decisions can be based on a knowledge of the distributions and observed data. It provides a means of reasoning about alternative hypotheses, given related data. Bayesian analysis can also help with the understanding of algorithms that do not explicitly manipulate probability based data.

Bayes theorem is useful in that it provides a method for assessing the likelihood of a given hypothesis given the available data and prior probabilities about the hypothesis and data distribution. Let $P(h)$ be the prior probability of the validity of the hypothesis h, $P(D)$ be the prior probability that the training data D will be observed. Then $P(D \mid h)$ is the probability of observing the data D in a world in which the hypothesis h is valid. Generally $P(X \mid Y)$ is the probability of X given Y. Machine learning is generally interested in $P(h \mid D)$, that is the probability of the hypothesis being valid given the observed data. This measure will allow a ML system to select from a number of competing hypotheses. Bayes theorem gives us a mechanism to make that selection.

Definition 17. Bayes Theorem is defined as:

$$P(h \mid D) := \frac{P(D \mid h)P(h)}{P(D)} \tag{2.40}$$

It is often the case in ML that we need to select the most likely hypothesis from some set of hypotheses. Bayes theorem allows us to choose the maximum *a-posteriori*, MAP, hypothesis, h_{MAP}.

Definition 18. The **maximum *a-posteriori* hypothesis**, h_{MAP} is defined as:

$$\begin{aligned} h_{MAP} &:= \arg\max_{h \in H} P(h \mid D) \\ &= \arg\max_{h \in H} \frac{P(D \mid h) P(h)}{P(D)} \\ &= \arg\max_{h \in H} P(D \mid h) P(h) \text{ since } P(D) \text{ is independent of h} \end{aligned} \tag{2.41}$$

where $\arg\max_{x \in X} f(x)$ is the value of x that maximises the value of $f(x)$.

Often it is the case that we assume all hypotheses are equally likely, see section 3.8.1 for further discussion of this point. Then the maximum likelihood hypothesis h_{ML}, would be

$$h_{ML} = \arg\max_{h \in H} P(D \mid h) \tag{2.42}$$

Bayesian learning has a number of useful features:

- Prior knowledge can be combined with observed data to determine the final probability of a hypothesis.
- Bayesian methods can accommodate hypotheses that make probabilistic predictions.
- Each observed training example can incrementally change the estimated probability that a given hypothesis is correct.
- New instances can be classified by combining the predictions of multiple hypotheses, weighted by their probabilities.
- It provides a standard of optimal decision making against which other methods can be measured.

Bayesian analysis is also able to provide support for the least squared error hypothesis, gradient descent and highlight the analogy between the minimum description length principle and the maximum *a-posteriori* hypothesis [Shannon & Weaver 1949].

Once the analysis of the hypotheses and data has been made and h_{MAP} identified, it might seem reasonable to use this hypothesis to classify a new instance. However, the work already performed in finding h_{MAP} makes possible another more accurate method of classifying new instances which makes use of all available hypotheses and information of their likelihood of being correct. This method of classification is known as the Bayes optimal classifier. This classifier combines the predictions of all the hypotheses to find the most probable classification.

Definition 19. Assuming the classification can take any value v_j from some set V, the the probability $P(v_j \mid D)$ that v_j is the correct classification is

$$P(v_j \mid D) = \sum_{h_i \in H} P(v_j \mid h_i) P(h_i \mid D) \tag{2.43}$$

Then the optimal classification of the new instance is just the value v_j which has the maximum $P(v_j \mid D)$. Thus the **Bayes optimal classification** is given by

$$v_{BayesOptimal} := \arg\max_{v_j \in V} \sum_{h_i \in H} P(v_j \mid h_i) P(h_i \mid D) \tag{2.44}$$

It can be shown that no other classification method can outperform the Bayes optimal classifier, given the same set of data, hypothesis space, and prior knowledge about the probability of hypotheses. The Bayes optimal classifier also has the property that it can represent a linear combination of the hypotheses in H and thus potentially a hypothesis not in H. Alas the draw back to this classification system is the amount of work required to obtain each classification. A computationally cheaper option is the Gibbs algorithm [Opper & Haussler 1991], which selects a hypothesis h from H at random based on its posterior probability distribution over H. It has been shown under certain conditions to have at most twice the expected error of the Bayes optimal classifier.

2.9.1 Naïve Bayes Classifier

The naïve Bayes classifier [Duda & Hart 1973], has proved to be a successful approach to using Bayes theorem for learning. The naïve Bayes classifier assumes that each instance x is described

by the conjunction of a set of attribute values, and that the target function $f(x)$ can take on any value from some finite set V. A set of training examples of the required target function is provided, and a new instance is presented, described by a tuple of attribute values $< a_1, a_2, \ldots, a_n >$. The learner has to predict the classification for this new instance. A Bayesian approach would be to supply the most probable value v_{MAP} given the attribute values $< a_1, a_2, \ldots, a_n >$ that describe the instance

$$v_{MAP} = \arg\max_{v_j \in V} P(v_j \mid a_1, a_2, \ldots, a_n) \tag{2.45}$$

using Bayes theorem we can rewrite this as

$$\begin{aligned} v_{MAP} &= \arg\max_{v_j \in V} \frac{P(a_1, a_2, \ldots, a_n | v_j) P(v_j)}{P(a_1, a_2, \ldots, a_n)} \\ &= \arg\max_{v_j \in V} P(a_1, a_2, \ldots, a_n \mid v_j) P(v_j) \end{aligned} \tag{2.46}$$

The naïve Bayes classifier then makes the following simplifying assumption, that the attribute values are conditionally independent given the target value. That is, given the target value, the probability of observing the conjunction of attributes $a_1, a_2, \ldots, a_n$ is just the product of the probabilities of the individual attributes: $P(a_1, a_2, \ldots, a_n \mid v_j) = \prod_i P(a_i \mid v_j)$. Then we have

$$v_{NB} = \arg\max_{v_j \in V} P(v_j) \prod_i P(a_i \mid v_j) \tag{2.47}$$

where v_{NB} denotes the target value output by the naïve Bayes classifier. Unlike many other forms of learning the naïve Bayes classifier does not appear to search a hypothesis space for a general solution. The values for its probabilities are simply those found in the training data. Despite its assumptions about independent variables, the naïve Bayes classifier is surprisingly robust when dealing with situations in which these assumptions are not correct [Domingos & Pazzani 1997].

There are difficulties in assigning probabilities to the values of continuous variables when the distribution of the variables is not known. In the case of a naïve Bayes classifier where the dataset includes continuous variables, either some distribution has to be assumed for the continuous variables, often the Gaussian distribution, a non-parametric method used to estimate probabilities, such as the kernel method discussed in [John & Langley 1995], or some method of discretisation must be used. Dougherty, [Dougherty et al 1995], and Kohavi and Sahami, [Kohavi & Sahami 1996], both show that sophisticated methods of discretisation can improve the performance of a naïve Bayes classifier when compared to something simple like “ten-bin”

which involves dividing a continuous variable into ten equal width bins. Some insight into the reasons for the success of naïve Bayes in dealing with continuous variables can be found in [Hsu, Huang & Wong 2000].

There are numerous developments of the naïve Bayes classifier which attempt to improve its accuracy. One method for doing this is to account for and minimise bias and variance introduced through discretisation of continuous variables [Yang and Webb 2009]. Another common theme is to allow for a weakening of the independence assumption while retaining the simplicity and efficiency of the naïve Bayes classifier. Of these developments Lazy Bayesian Rules, LBR, [Zheng & Webb 2000], Super Parent TAN, SP-TAN, [Keogh & Pazzani 1999], a development of tree-augmented naïve Bayes, TAN, [Friedman & Goldszmidth 1996], and aggregating one-dependence estimators, AODE, [Webb et al 2005] with developments Gaussian AODE, GAODE, and hybrid AODE, HAODE, [Flores et al 2009] are among the most promising. They deliver increased accuracy with weakening of the independence requirements, but at the cost of increased computational complexity. Another approach introduces latent variables as a way of capturing the dependencies between the attributes [Langseth & Nielsen 2005]. Although naïve Bayes classifiers can deal directly with multi-class classification, there is some work on determining if decomposing such a multi-class problem into a number of binary classification problems can improve the learning accuracy [Sulzmann et al 2007].

Bayesian methods in general and the naïve Bayes classifier in particular are used extensively with many ML learners for a variety of purposes including, but not limited to: model selection [Rusakov & Geiger 2005]; text classification [McCallum and Nigam 1998, Frank & Bouckaert 2006, Kim et al 2006]; relational learning[10] [Landwehr et al 2007]; ranking results [Zhang et al 2005]; and numerous other uses.

2.9.2 The Expectation-Maximisation (EM) algorithm

There are situations when some variables cannot be observed on all occasions or at all. An approach called the EM algorithm [Dempster et al 1977] can be used to learn in the presence of unobserved variables. It is possible to use the EM algorithm even when there are variables whose values are never directly observed provided the form of the probability distribution governing the variables is known. Let $X = \{x_1, \ldots, x_m\}$ denote the observed data in a set of m independently drawn instances, let $Z = \{z_1, \ldots, z_m\}$ denote the unobserved data in the same instances and let

[10] This term is used when the learning combines probabilistic and logical elements.

$Y = X \cup Z$ denote the full data. The unobserved data Z can be treated as random variables whose probability distribution depends on the unknown parameters θ and on the observed data X. Y is a random variable because it is defined in terms of the random variable Z. Let h denote the current hypothesised values of the parameters θ, and h' denote the revised hypothesis that is estimated on each iteration of the EM algorithm. The EM algorithm searches for the maximum likelihood hypothesis h', by seeking the h' that maximises $E[\ln P(Y \mid h')]$. This expected value is taken over the probability distribution governing Y, which is determined by the unknown parameters θ. Given that the full data Y is a combination of the observed data X and the unobserved data Z, we must average over the possible values of the unobserved Z, weighing each according to its probability. That is, we take the expected value $E[\ln P(Y \mid h')]$ over the probability distribution governing the random variable Y. The distribution governing Y is determined by the known values for X, plus the distribution governing Z. In general the distribution governing Y will be unknown as it is determined by the parameters θ that we are trying to estimate. So, the EM algorithm uses is current hypothesis h in place of the actual parameters θ that we are trying to estimate. Let us define a function $Q(h' \mid h)$ that gives $E[\ln P(Y \mid h')]$ as a function of h', under the assumption that $\theta = h$ and given the observed portion X of the full data Y. Then

$$Q(h' \mid h) = E[\ln P(Y \mid h') \mid h, X] \tag{2.48}$$

and in its general form the EM algorithm is given in algorithm 2.10. When the function Q is

Algorithm 2.10 General Form of the EM Algorithm

1. Repeat until termination conditions are met

   (a) Estimation (E) Step: Calculate $Q(h' \mid h)$ using the current hypothesis h and the observed data X to estimate the probability distribution over Y.

   $$Q(h' \mid h) \leftarrow E[\ln P(Y \mid h') \mid h, X]$$

   (b) Maximisation (M) Step: Replace hypothesis h by the hypothesis h' that maximises this Q function.

   $$h \leftarrow \arg\max_{h'} Q(h' \mid h)$$

continuous, the EM algorithm converges to a stationary point of the likelihood function $P(Y \mid h')$. When the likelihood function has a single maximum, EM will converge to this global maximum likelihood estimate for h'. Otherwise it is guaranteed only to converge to a local maximum. In

this respect EM shares some of the limitations of other optimisation methods such as gradient descent.

Like naïve Bayes the EM algorithm is used in many different circumstances, and varieties [Neal and Hinton 1998], anywhere there are missing data items, including: general optimisation [Salakhutdinov et al 2003], reinforcement learning [Hachiya et al 2009], analysing causal relationships [Jugelenaite & Heskes 2006], working with Gaussian mixtures [Zhang et al 2008], as a parallel algorithm for large datasets [Wolfe Haghight & Klein 2008], and for text classification [Nigam et al 2000].

2.9.3 Bayesian Networks

A Bayesian belief network, BBN or BN, [Pearl 1988, Heckerman 1995a] describes the probability distribution governing a set of variables by specifying a set of conditional independence assumptions along with a set of conditional probabilities. Bayesian belief networks represent conditional independence assumptions that apply to subsets of the variables. In general a Bayesian belief network describes the probability distribution over a set of variables. Consider an arbitrary set of random variables $Y_1, Y_2, \ldots, Y_n$, where each variable Y_i can take on the set of possible values $V(Y_i)$. We define the joint space of the set of variables Y to be the cross product $V(Y_1) \times V(Y_2) \times \cdots \times V(Y_n)$. Each item in the joint space corresponds to one of the possible assignments of values to the tuple of variables $< Y_1, \ldots, Y_n >$. The probability distribution over this joint space is called the joint probability distribution. The joint probability distribution specifies the probability of each of the variable bindings for the tuple $< Y_1, \ldots, Y_n >$. A Bayesian belief network, describes the joint probability distribution for a set of variables.

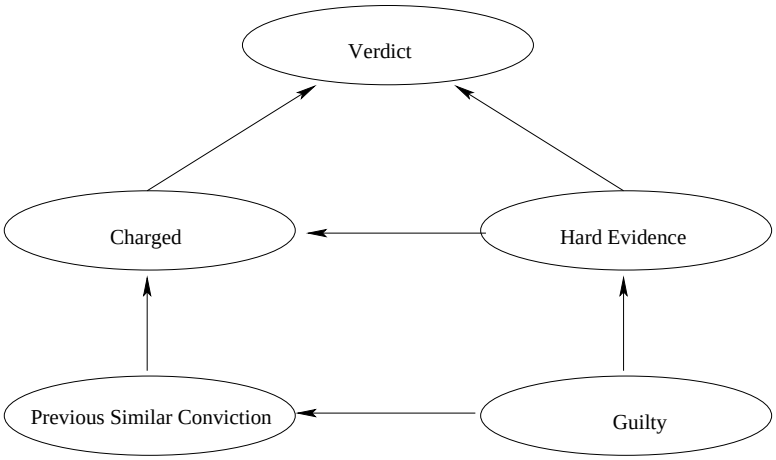


Figure 2.9: A Bayesian Belief Network.

The diagram in figure 2.9 represents the joint probability distribution over the boolean variables, `Verdict, Charged, Hard Evidence, Previous Similar Conviction`, and `Guilty`. The table 2.10 represents the `Charged` node's conditional probability table,

	Hard Evidence	Yes		No	
	Previous Similar Convictions	Yes	No	Yes	No
Yes		0.9999	0.99	0.02	0.00001
No		0.0001	0.01	0.98	0.99999

Figure 2.10: Conditional Probability Table for the *Charged* Node

sometimes called its node probability table, NPT. In general a BN represents the joint probability distribution by specifying a set of conditional independence assumptions (represented by a directed acyclic graph, DAG), together with sets of local conditional probabilities. Each variable in the joint space is represented by a node in the BN. For each variable two types of information are specified: the network arcs represent dependency between joined variables; and a conditional probability table is given for each variable, describing the probability distribution of that variable given the value of its immediate predecessors. The **Markov condition** that each variable is conditionally independent of its non-descendants in the network given its immediate predecessors in the network is assumed to be true. The joint probability for any desired assignment of values $< y_1, y_1, \ldots, y_n >$ to the tuple of network variables $< Y_1, Y_2, \ldots, Y_n >$ can be computed by the formula

$$P(y_1, \ldots y_n) = \prod_{i=1}^{n} P(y_i \mid Parents(Y_i)) \tag{2.49}$$

where $Parents(Y_i)$ denotes the set of immediate predecessors of Y_i in the network. The values of $P(y_i \mid Parents(Y_i))$ are the values of the conditional probability tables associated with node Y_i. Conditional independencies within a DAG can be discovered using a property called d-separation defined in [Pearl 1988][11]. BNs are a convenient way to represent causal knowledge such as how hard evidence against someone will lead to their being charged [Fenton and Neil 2000]. It is worth noting that the network diagram shown in figure 2.9 does not represent a true causal network since `Guilty` and `Previous Similar Conviction`, while linked in the network,

[11] D-separation defines the class of conditional independencies entailed by the Markov condition.

are not causally linked. BNs are often used in decision making, it's important when doing so to understand the stability of the decision to changes in the data, this is often called the sensitivity of the BN to changes in the data. Gaag and Coupé provide one method of performing this analysis [Van Der Gaag & Coupé 1999]. It is also important to understand the uncertainty in a BN when making decisions. In multicriteria decision making understanding the uncertainties and limitations of a BN can lead to improved decision making [Fenton & Wang 2006]. When dealing with a sequence of data, such as a time series, a modified form of BN called a dynamic BN can be used to model the sequence [Kjærulff 1992, Ghahramani 1998]. BNs provide a powerful form of representation for many problems, however, is it not easy to get an intuitive idea of the complexity of target function which can be represented by a given BN. Using the XOR function as a unit of complexity Ling and Zhang, [Ling & Zhang 2002], prove that, approximately, a BN in which each node has no more than, k, parents cannot represent a target function with $(k+1)$XORs.

BNs allow us to infer the probability distribution of a node given the observed value of the other nodes. In the more general case we may wish to infer the probability distribution for some node given values for only a subset of the other nodes. In general a BN can be used to compute the probability distribution for any subset of nodes given the values or distributions for any subset of the remaining nodes. Exact inference of probabilities in general for an arbitrary BN is known to be NP-hard [Cooper 1990]. Even approximate inference of probabilities in a BN can be NP-hard [Dagum & Luby 1993], but approximate methods have been shown to be useful in many cases. In cases when a BN is required to provide a timely response to updated information there is some work to identify the parts of the BN that are most important to the variables of interest, thus allowing for inference over a sub-network. Such work usually has some measure of the influence of one or more variables on another [Boerlage 1994, Nicholson & Jitnah 1998].

There is a significant amount of research into learning BNs from data. The problem varies depending on how much is known at the outset. If the network structure is known and all the variables are visible then a naïve Bayesian classifier can be used to deduce the node probability tables from the data. If the structure is given, but some of the variables are not visible in the available data then the problem is similar to learning the weights for the hidden nodes in an ANN. [Russel et al 1995] propose a similar gradient ascent procedure that learns the probability tables.

One view of the algorithms that attempt to learn BNs from data is that they can be divided into two broad classes; those that attempt to learn the entire network in one go, and so have some method of ranking or scoring the candidate networks, these are usually called search-and-score, the alternative method, called constraint-based attempts to build the network one piece at a time using a statistical or information theoretic tests compared to some threshold. Since a BN can be viewed as a causal network, with the parents acting as direct causes of their children, building the structure of a BN becomes a causal discovery process.

The Bayesian approach to learning a complete network structure, as used in [Heckerman et al 1995] attempts to look for the *most probable*, in a Bayesian sense, structure given the data. Since a BN is the target type of structure, only DAGs are usually considered. This process usually has to be constrained in some way as it is otherwise computationally infeasible [Chickering et al 1994]. [Cooper & Herskovits 1992] present a Bayesian scoring metric and heuristic search algorithm called K2, which can be used to infer BNs from data when the network structure is not known. In the case of K2 an additional constraint is supplied in the form of a node ordering which determines the parent child relationship between any two related nodes.

Constraint-based techniques vary, but most assume the Causal Markov condition.

> **Definition 20.** The **Causal Markov Condition** states that, any node in a Bayesian network is conditionally independent of its non-effects, given its direct causes.

The parents of a node are its direct causes. This requirement restricts the network structure to a DAG, which is what would be expected of a BN anyway. While this would not appear to be a poor definition of causal relationships, it does preclude causal loops. The learner will attempt to construct an appropriate DAG by either adding, as in the IC algorithm [Pearl and Verma 1991], or removing [Spirtes et al 1993] edges as required. Constraint-based genetic algorithms have also been used to determine BN structure [Larranaga et al 1996]. The constraint-based learners usually have some form of conditional independence test. One difficulty of this is that such tests can be very computationally expensive when dealing with large datasets. An approach to this problem used in [Yehezkel & Lerner 2009] is to use relatively cheap tests to identify autonomous sub-structures within the DAG then to use more costly tests to accurately determine the relationships within the sub-structure. A similar strategy is used by [Margaritis & Thrun 2000] which uses Markov blankets (see [Pearl 1988]) to identify a local neighbourhood of each variable and then searches for relationships within a bounded neighbourhood size.

Given some knowledge of the structure of all or part of the BN, additional knowledge can be used to constrain the relationships within it as shown by [Niculescu et al 2006, Perrier et al 2008]. In [Tsamardinos et al 2006] a combined approach using local learning, constraint-based, and search-and-score techniques is used in an algorithm called, Max-Min Hill-Climbing, MMHC, which appears to be competitive with other techniques of structure discovery for BNs. There is also research into learning dynamic BNs from data [Ghahramani 1998].

In some instances a DAG is not appropriate as a representation of a domain, directed mixed graphs, DMGs, are a more general representation than DAGs and also allow for conditional independence testing using a concept of m-separation [Richardson 2003]. [Silva & Ghahramani 2009] show how DMGs can be used to eliminate some complexities caused by hidden variables, modelling the dependencies caused by the hidden variables without actually modelling the hidden variables. However, hidden variables are not always detrimental, in cases where a conditional independence indicated by the network is absent, adding a hidden variable, [Kwoh & Gillies 1996], can improve the performance of the network and, presumably, its accuracy as a model for the underlying system.

While all the methods of causal discovery from data rely on some statistical tests, [Cooper 1997] demonstrates a relatively simple use of statistical tests which can determine certain limited causal relationships, but is computationally feasible on large complex datasets. This work is extended by [Silverstein et al 2000] to cover additional causal tests and these tests are adapted and further extended in this work, see chapter 5.

BNs are very useful for representing domain knowledge and causal relationships. The structure of a BN can be considered as a direct representation of, at least part of, the underlying causal structure of the domain being represented. Similarly the NPTs represent the type and strength of the influence of the causes, parent nodes, on their effects, child nodes. This is because the NPT specifies how the values of the parents determine the distribution of the values of the child. So the NPT shows how changes in the value of any given parent will effect the distribution of values of the child. Available domain knowledge can be used to help determine both the structure and NPTs of a BN, but potentially either can be supplied or learnt from data if necessary.

2.10 Support Vector Machines (SVMs)

Support Vector Machines, SVMs, are kernel-based learners and while most of the ideas used in SVMs have been around some years the paper [Boser, Guyon & Vapnik 1992] brought the ideas together. SVMs are an example of a *kernel* machine, KM. KMs are a class of algorithm for pattern analysis. KMs use a *kernel* function [Mercer 1909, Aizerman, Braverman & Rozonoér 1964] which transforms its input data into the inner product of a higher dimensional normed vector space, the feature space.

Definition 21. The **inner (dot) product** of two vectors $\bar{x}$ and $\bar{y}$ is defined as

$$(\bar{x} \cdot \bar{y}) := \sum_i x_i y_i \tag{2.50}$$

where x_i and y_i are the ith components of the vectors $\bar{x}$ and $\bar{y}$ respectively.

Definition 22. If X is the input space and $\mathcal{H}$ is the feature space we define a map $\Phi : X \to \mathcal{H}$ then we can define the **kernel function** k as

$$k(\bar{x}, \bar{y}) := (\bar{X} \cdot \bar{Y}) = (\Phi(\bar{x}) \cdot \Phi(\bar{y})) \tag{2.51}$$

It is possible to determine if a function is a *kernel* function [Mercer 1909].

SVMs are linear learning machines. In the simplest case of classification the data can be divided into two groups by a hyperplane, the learning problem is then simply that of finding the optimal hyperplane. The optimal hyperplane, is one that bisects and is orthogonal to the shortest line joining the convex hulls of the two classes, $+$ and $-$ in the example shown in figure 2.11
In figure 2.11 above the support vectors are circled. The input space can be linearly separated using:

$$\begin{aligned} (\bar{w} \cdot \bar{x}) + b &= 0, b \in \mathbb{R} \\ f(\bar{x}) &= (\bar{w} \cdot \bar{x}) + b \\ h(\bar{x}) &= sgn(f(\bar{x})) \end{aligned} \tag{2.52}$$

where $h(\bar{x})$ classifies instances into $+$ or $-$. Consider the case of the perceptron [Rosenblatt 1957] where the target space is $y \in Y = \{1, -1\}$. We then have

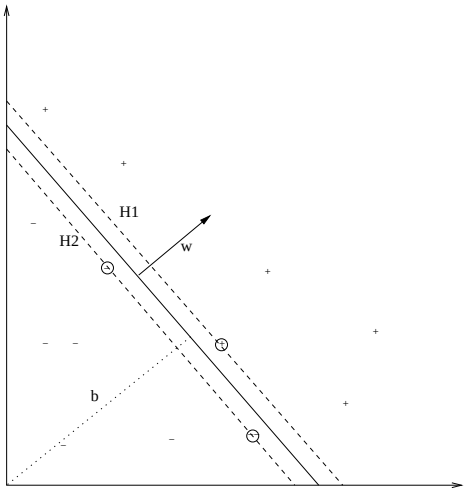


Figure 2.11: Hyperplane Separating + and - Items

$$\begin{aligned}(\bar{x}\cdot\bar{w})+b &\geq 1 \text{ for } y=1\\ (\bar{x}\cdot\bar{w})+b &\leq -1 \text{ for } y=-1\end{aligned} \tag{2.53}$$

combining these we get

$$\forall i : y_i\left((\bar{x}_i\cdot\bar{w})+b\right)\geq 0 \tag{2.54}$$

In figure 2.11 the line H2 is defined by

$$(\bar{x}\cdot\bar{w})+b=1 \tag{2.55}$$

and similarly H1 is defined by

$$(\bar{x}\cdot\bar{w})+b=-1 \tag{2.56}$$

The perpendicular distance from H2 to the origin is $|1-b|/\|\bar{w}\|$ and for H1 its $|-1-b|/\|\bar{w}\|$. So the distance between H1 and H2, the margin, is $2/\|\bar{w}\|$. To find the optimal hyperplane, the one with the maximal margin of separation between H1 and H2 (there are no training points between H1 and H2) we can minimise $\|\bar{w}\|^2$. Constructing the optimal hyperplane is done by optimising

$$\begin{aligned}&\min_{\bar{w},b}\frac{1}{2}\|\bar{w}\|^2\\ &\quad\text{keeping}\quad y_i\left((\bar{w}\cdot\bar{x}_i)+b\right)\geq 0\end{aligned} \tag{2.57}$$

The update rule then becomes:

$$\begin{aligned} \text{if } y_i(\bar{w}_k \cdot \bar{x}_i) &\leq 0 \text{ then} \\ \bar{w}_{k+1} &\leftarrow \bar{w}_k + \eta y_i \bar{x}_i \\ k &\leftarrow k+1 \end{aligned} \tag{2.58}$$

(where η is a positive constant analogous to the learning rate in other machine learning algorithms). This generates the required vector $\bar{w}$. This algorithm produces a result

$$\begin{aligned} \bar{w} &= \sum_i \alpha_i y_i \bar{x}_i \\ \alpha_i &\geq 0 \end{aligned} \tag{2.59}$$

which is a linear combination of the training points $\bar{x}_i$. Only points which would cause a misclassification effect the update rule. The function $f(\bar{x})$ can now be rewritten as

$$\begin{aligned} f(\bar{x}) &= (\bar{w} \cdot \bar{x}) + b \\ &= \sum_i \alpha_i y_i (\bar{x}_i \cdot \bar{x}) + b \end{aligned} \tag{2.60}$$

This is called the *dual representation* for SVMs. So the update rule becomes:

$$\text{if } y_i \left(\sum_j \alpha_j y_j (\bar{x}_j \cdot \bar{x}_i) + b \right) \leq 0 \text{ then } \alpha_i \leftarrow \alpha_i + \eta \tag{2.61}$$

In this linear learning machine, one important consideration is that the input data now only appears as part of an inner product, in both the learning rule and decision function, $f(\bar{x})$. This learner can only deal with linearly separable non-noisy data. A solution when dealing with data which is not linearly separable is to map the input data to a new feature space with non-linear features. If a good mapping is chosen the data will be linearly separable in the new feature space. However, the representation of such a feature space may be complex and an increase in dimensionality and/or vector size leads to increasing computational complexity. If we have a mapping $\bar{x} \rightarrow \Phi(\bar{x})$ then the decision function becomes

$$f(\bar{x}) = \sum_i \alpha_i y_i (\Phi(\bar{x}_i) \cdot \Phi(\bar{x})) + b \tag{2.62}$$

In figure 2.12 we see how an initially non-linear problem can be transformed into a linear one by the use of an appropriate transformation Φ. However, simply having an appropriate transformation may not be enough as the new feature space could be computationally difficult to work with,

or require a very large, possibly infinite, representation. Such problems could mean that while an appropriate mapping exists the learning process is difficult to compute. Recall that a *kernel*

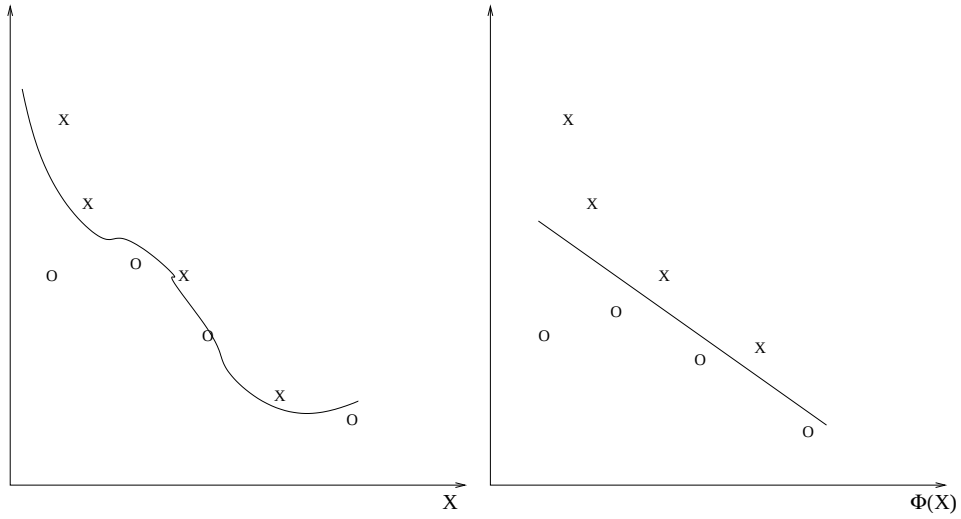


Figure 2.12: Transforming a Non-Linear Problem to a Linear One

function, k, is defined by $k(\bar{x},\bar{y}) := (\bar{X}\cdot\bar{Y}) = (\Phi(\bar{x})\cdot\Phi(\bar{y}))$. So by rewriting the learning rule and decision function using a *kernel* function we get

$$\begin{aligned} f(\bar{x}) &= \sum_i \alpha_i y_i (\Phi(\bar{x}_i)\cdot\Phi(\bar{x})) + b \\ &= \sum_i \alpha_i y_i (k(\bar{x}_i,\bar{x})) + b \end{aligned} \tag{2.63}$$

In this new representation there is no need to represent the transformed vector space, it isn't even required that the transformation is known. As long as, k, is a *kernel* function the transformation can be assumed to exist. This use of the dual representation and the *kernel* trick [Aizerman, Braverman & Rozonoér 1964] allows the use of a transformed feature space without any additional representational complexity. Thus it allows SVMs to work with non-linear problems. It is worth noting that since the kernel function effectively defines the feature space, it determines the types of relationships that can be found in the data. So, choosing an optimal kernel function is aided by knowledge of the types of relationships expected in the data. The optimal hyperplane is found with

$$\begin{aligned} \max_{\bar{\alpha}} W(\bar{\alpha}) &= \sum_i \alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j k(\bar{x}_i,\bar{x}_j) \\ \text{keeping} \quad \alpha_i \geq 0, &\quad \sum_i \alpha_i y_i = 0 \end{aligned} \tag{2.64}$$

A modified version of SVMs, Transductive Support Vector Machines, TSVMs [Vapnik 1998], was proposed which allows an SVM to take into account known structural properties of the data. However the results of TSVMs can be unstable [Wu et al 1999] and there is some uncertainty in the underlying theory for TSVMs [Zhang and Oles 2000].

The SVMs covered so far will only be able to find an optimal hyperplane in those cases where no data items are mislabelled and any noise in the data does not cause overlapping classes. An extension to SVMs called the Soft Margin Method [Cortes and Vapnik 1995] provides a way of dealing with these problems. New variables called slack variables, ξ_i, are used to measure the degree of misclassification of a data point $\bar{x}_i$. An optimal hyperplane is then found by optimising

$$\begin{aligned} \min_{\bar{w},b} \frac{1}{2}\|\bar{w}\|^2 & \\ \xi_i \geq 0 & \\ \text{keeping} \quad y_i\left((\bar{w}\cdot\bar{x}_i)+b\right) & \geq 1-\xi_i \end{aligned} \tag{2.65}$$

A classifier with good generalisation properties can be found by controlling capacity (via $\|\bar{w}\|$) and the sum of the slack variables $\sum_i \xi_i$. C-SVC is an example of a soft margin classifier which minimises the objective function

$$\tau\left(\bar{w},\bar{\xi}\right) = \frac{1}{2}\|\bar{w}\|^2 + C\sum_i \xi_i \tag{2.66}$$

where $C > 0$. This once again has a solution of the form $\bar{w} = \sum_i \alpha_i y_i \bar{x}_i$ with the modified constraint $0 \leq \alpha_i \leq C$.

SVMs have been used in many areas of research including genetics [Park et al 2007], text categorisation [Joachims 2002], optimisation [Eitrich and Lang 2006], neural networks [Lin et al 2005] and many other fields. There are numerous extensions to the basic SVM including learners with the option to abstain from making a classification [Grandvalet et al 2009], limiting the number of support vectors [Dekel and Singer 2007], parallel processing variations [Graf et al 2005, Chang et al 2008], and many other modifications to the basic SVM. Other recent work has demonstrated that it is possible to use SVMs to determine independence between variables in datasets with the use of unconditional and conditional cross-covariance operators in reproducing kernel Hilbert spaces [Scholkopf & Smola 2002], and together with the logic from other causal learners like IC and K2, see chapter 4, to form the basis of causal discovery from data [Sun et al 2007].

Since choosing an optimal kernel function is crucial to the functioning of an SVM there is active research into both how best to use existing knowledge in choosing a kernel function and into learning a kernel function which will perform well on data with limited background knowledge or expert input. Research into learning kernels when additional information is available, either in the form of labeled examples or pairwise constraints, where the constraint defines if the pair belong in the same or different classes, has yielded promising results [Kondor and Lafferty 2002, Chapelle et al 2003, Kulis et al 2006, Zhu et al 2005]. There is also research which aims to improve on the usefulness and/or reduce the amount of additional data that is needed by selecting the most informative data pairs whose classification can then be used in learning an optimal kernel [Hoi and Jin 2008].

An alternative is learning kernels with prior knowledge expressed as linear constraints, see [Mangasarian et al 2004], and this can be further extended to non-linear constraints examined in [Mangasarian and Wild 2007]. If additional information is known about the derivative of the target function or bounds on it, in general or at specific points, then further improvements on the SVM can be made [Lauer and Bloch 2007]. It is also possible to use domain knowledge to explain the classification of a given example [Sun & DeJong 2005]. These methods have all demonstrated that prior knowledge can be used to improve the accuracy of SVMs.

2.11 Explanation-Based Learning

Explanation-based learning, like ILP is an analytical form of learning where the aim is to produce one or more hypotheses that are consistent with both the data and the domain knowledge. It is simplest to consider explanation-based learning with perfect domain theories, that is, domain theories that are correct and complete. A domain theory is said to be correct if each of its assertions is a truthful statement about its world. A domain theory is said to be complete with respect to a given target concept and instance space, if the domain theory covers every positive example in the instance space. It is usually assumed that the PROLOG convention that anything which cannot be proven true is assumed false is applied here. So, the definition will suffice for both positive and negative examples. A general introduction and overview can be found in [DeJong 2004].

A good example of explanation-based learning is the sequential covering algorithm PROLOG-EBG [Kedar-Cabelli & McCarty 1987]. When supplied with complete and correct domain infor-

mation, PROLOG-EBG will produce a correct hypothesis that covers the positive training examples. The output hypothesis (set of rules) of PROLOG-EBG is a set of logically sufficient conditions for the target concept, given the domain theory. The PROLOG-EBG algorithm is shown in algorithm 2.11. When analysing the explanation PROLOG-EBG uses the idea of the

Algorithm 2.11 The PROLOG-EBG Algorithm

1. PROLOG-EBG($TargetConcept, TrainingExamples, DomainTheory$)
2. $LearnedRules \leftarrow \{\}$
3. Pos$\leftarrow$ the positive examples from $TrainingExamples$
4. for each $PositiveExample$ in Pos that is not covered by $LearnedRules$ do
   (a) Explain: $Explanation \leftarrow$ an explanation (proof) in terms of the $DomainTheory$ that $PositiveExample$ satisfies the $TargetConcept$
   (b) Analyse: $SufficientConditions \leftarrow$ the most general set of features of $PositiveExample$ sufficient to satisfy the $TargetConcept$ according to the $Explanation$
   (c) Refine: $LearnedRules \leftarrow LearnedRules + NewHornClause$, where $NewHornClause$ is of the form
   $$TargetConcept \leftarrow SufficientConditions$$
5. Return $LearnedRules$

weakest preimage.

> **Definition 23.** This can be defined as follows: The **weakest preimage** of a conclusion C with respect to a proof P is the most general set of initial assertions A, such that A entails C according to P.

PROLOG-EBG computes the weakest preimage using a general procedure called regression [Waldinger 1977]. Regression operates on the set of Horn clauses which compose the domain theory. It works iteratively backwards through the explanation, first computing the weakest preimage of the target concept with respect to the final proof step in the explanation, then computing the weakest preimage of the resulting expressions with respect to the preceding step and so on. The procedure terminates when it has iterated over all the steps in the explanation, yielding the weakest preimage of the target concept with respect to the literals at the leaf nodes of the explanation. The regression algorithm is outlined in algorithm 2.12.

PROLOG-EBG uses the domain knowledge together with the examples to justify the hypothesis it produces. The explanation of how an individual example satisfies the target concept

Algorithm 2.12 Algorithm for Regressing a Set of Literals through a single Horn Clause

1. REGRESS($Frontier, Rule, Literal, \theta_{hi}$)
2. $Frontier$: Set of literals to be regressed through $Rule$
3. $Rule$: A Horn clause
4. $Literal$: A literal in $Frontier$ that is inferred by $Rule$ in the explanation
5. θ_{hi}: The substitution that unifies the head of $Rule$ to the corresponding literal in the explanation
6. Returns the set of literals forming the weakest preimage of $Frontier$ with respect to $Rule$
   (a) $head \leftarrow$ head of $Rule$
   (b) $body \leftarrow$ body of $Rule$
   (c) $\theta_{hl} \leftarrow$ the most general unifier of $head$ with $Literal$ such that there exists a substitution θ_{li} for which
   $$\theta_{li}(\theta_{hl}(head)) = \theta_{hi}(head)$$
   (d) Return $\theta_{hl}(Frontier - head + body)$

determines the relevance of its attributes. Those attributes which are required as part of the explanation are precisely those which are relevant. Each learnt Horn clause corresponds to a sufficient condition for satisfying the target concept. The set of Horn clauses covers the positive training examples encountered by the learner, and any others which share the same explanation. The generality of the learnt Horn clauses will depend on both the formulation of the domain theory, and the order in which the training examples are considered. There is an implicit assumption that the domain theory is correct and complete and that the examples do not contain any noise.

An interesting property of PROLOG-EBG is that it can discover new features, that is, from the combination of the domain knowledge and training examples it can find features which characterise part of the target concept. This is similar to the feature discovery in the hidden layers of ANNs. One significant difference is that any such features are easily visible as part of the explanations that PROLOG-EBG builds. It is not easy to determine the exact inductive bias of a learner like PROLOG-EBG. However, a good approximation to that bias is the domain theory, and a preference for small sets of maximally general Horn clauses.

One area where the restrictions of correct and complete domain knowledge can easily be met is that of learning to speed up complex search programs. The largest scale attempts to use explanation-based learning have been in this field of 'speedup' learning. In practice many scheduling and optimisation problems can be formulated as large search problems. The PRODIGY

system [Carbonell et al 1990], uses explanation-based learning to improve its search. [Minton 1988] reports on experiments in three problem domains, in which the learnt control rules significantly improve problem solving efficiency. Like most forms of ML, EBL can be combined with other ML techniques to improve the overall performance of a system rather than to just act to improve speed. [Morik et al 1999] show how together with a support vector, feedback and an expert, EBL was able to improve the performance of a complex system and led to the discovery that critical information was missing from the domain rules. Similarly [Bay et al 2002] show how EBL can be used to improve a mathematical model when given an initial model and domain knowledge. [Sun & DeJong 2005] show how EBL can be used with a SVM to explain example classification.

EBL makes obvious good use of domain knowledge, however, when the domain knowledge is not complete and/or correct the tendency is to move towards a more ILP based approach [Pazzani 1989]. Recent developments in EBL have served to broaden its applicability. [DeJong 2006] shows how inference can be combined with consistency checks, an approach which seeks to maintain the advantages of exact deductive inference while adding information from observation into the learning process. A different direction is explained in [Kimming et al 2007], which incorporates uncertainty into both the domain knowledge and the learning process by using a probabilistic logic, ProbLog, rather than standard first-order logic. This form of knowledge, and inference, allows reasoning by analogy and so potentially simplifying knowledge discovery. Causation within an EBL learner is dependent on the form of the domain knowledge. Since EBL learners reason within the supplied domain knowledge, causation can either be clearly expressed or completely hidden depending on the design of the representation used for the domain knowledge.

2.12 Genetic Algorithms & Programs

The success of biological systems in adapting to changing conditions and exploiting a wide range of environments has led people to look at the possibility of using similar mechanisms for adaptation and *learning* in the field of machine learning. Genetic Algorithms, GAs, and Genetic Programs, GPs, are the result of this research effort. GAs and GPs use mechanisms analogous to crossover and mutation in genetic reproduction.

GAs typically work by starting with some random population of hypotheses. A fitness assessment is made of the population and a number of the fittest hypotheses are then carried over

intact, crossed and/or mutated to form the next generation of the population. In general the selection, crossover and mutation processes are probabilistic. The hypotheses are often encoded as bit strings. The interpretation of the bit string is dependent on the application. A typical genetic algorithm is shown in algorithm 2.13. In the algorithm: *Fitness* is a function which assigns an evaluation score to a given hypothesis; *Fitness_threshold* is a threshold specifying the termination criterion; p is the number of hypotheses to be included in the population; r is the fraction of the population to be replaced by crossover at each step; and m is the mutation rate.

Algorithm 2.13 A Typical Genetic Algorithm

1. GA($Fitness, Fitness\_threshold, p, r, m$)
2. Initialise the population: $P \leftarrow$ Generate p hypotheses at random
3. Evaluate: For each h in P, compute $Fitness(h)$
4. While $[\max_h Fitness(h)] < Fitness\_threshold$ do
   (a) Create a new generation, P_s
      i. Select: Probabilistically select $(1-r)p$ members of P to add to P_s. The probability $Pr(h_i)$ of selecting hypothesis h_i from P is given by
      $$Pr(h_i) = \frac{Fitness(h_i)}{\sum_{j=1}^{p} Fitness(h_i)}$$
      Crossover: Probabilistically select $\frac{r \cdot p}{2}$ pairs of hypotheses from P, according to $Pr(h_i)$. For each pair $< h_1, h_2 >$, produce two offspring by applying the Crossover operator. Add all offspring to P_s.
      ii. Mutate: Choose m percent of the members of P_s with uniform probability. For each invert one randomly selected bit in its representation.
      iii. Update: $P \leftarrow P_s$
      iv. Evaluate: for each h in P, compute $Fitness(h)$
5. Return the hypothesis from P that has the highest fitness

GAs can in theory use any form of encoding for the hypotheses. In practice bit strings are a common encoding as they allow relatively easy definitions of crossover and mutation operators. Suppose we want to represent an attribute with n possible values. Consider the case of a concept learner. We could use a string of n bits for one attribute, where each bit represented one of the possible attribute values. Then a 1 in the relevant bit would indicate the value was allowed and a 0 would indicate that value was disallowed. These bit strings for different attributes could then be grouped together to form a complete hypothesis. So, in our hypothesis string all 1s would be the most general hypothesis indicating that any value for any attribute was accept-

able and all 0s would be the most specific hypothesis indicating that no values were acceptable. More detailed examples of bit string representations in GAs can be found in [Holland 1986] and [DeJong et al 1993].

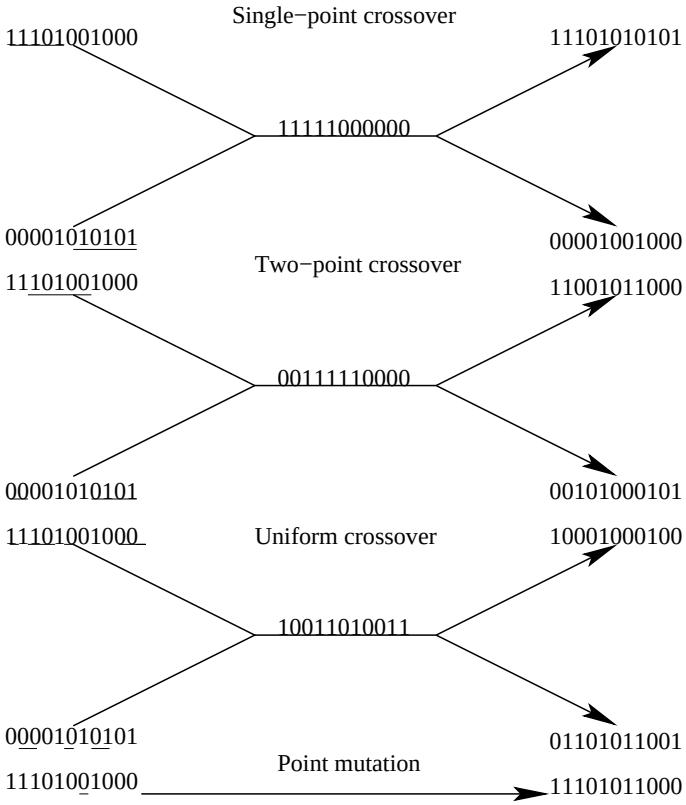


Figure 2.13: Some Operators for Genetic Algorithms

There are two major classes of operator which are central to genetic algorithms, crossover and mutation operators[12]. The crossover operators take two parent strings and a control string called the crossover mask, and produces two offspring. Each of the offspring will get a bit at any position from one of the parents with the other offspring getting the bit for that position from the other parent. In this fashion each bit from each parents is used in one of the offspring. There are many variations on the crossover operator that largely deal with how many times the offspring will change which parent they get the next bit from. Some common crossover operators are shown in figure 2.13. The mutation operator is a simple operator that simply switches one bit at random within the bit string. The mutation operator has a single parent and produces a single offspring. There can be variations on the mutation operator that change more than one bit.

[12]There are numerous other possible operators like inversion which while fairly common are not considered fundamental.

The above explanations are oversimplifications as, in general, crossing at any point or changing any single bit might produce a string that was meaningless. So some means of ensuring the bit string is meaningful after it has been altered will also be required. Other variations of the crossover operation split the original strings and simply swap the two right hand portions. This form of the crossover operator changes the length of the bit strings, and hence the complexity of the hypothesis they can represent. It is possible for GAs to have additional operators specific to the system or task. [Grenfenstette 1991] and [Janikow 1993] both have task specific operators which deal with specialising and or generalising learnt rule sets.

The selection of a member of a population of hypotheses for the next generation is influenced by its fitness rating, determined by the fitness function. The fitness function will usually assess the hypothesis's performance and may also take into account other properties of the hypothesis such as its complexity and generality. In general the selection method does not simply select the highest rated hypotheses, but uses some random selection biased by the performance of the hypotheses. The reason for this is that at any point in time it is likely that the best performing hypotheses are closely related. So, simply choosing the best performing hypotheses would reduce the variation within the population which both decreases the rate of evolution and increases the possibility of getting stuck at a performance maximum local to this particular hypothesis family. The algorithm shown in algorithm 2.13 uses what is called fitness proportionate selection. Other common selection methods are rank selection; in which the hypotheses are first ranked in order of fitness and then the ranking is used to determine selection probability; and tournament selection, in which hypotheses are chosen in pairs at random, then the more fit of the two is selected with some probability p or the less fit with a probability $(1-p)$. The problem of having too many similar individuals in the population is called crowding. There are a number of techniques which aim to reduce the likelihood of crowding, by methods such as sharing fitness between similar individuals, only allowing similar individuals to cross, and spatially separating individuals and then only allowing adjacent individuals to cross.

An examples of a GA system is the GABIL [DeJong et al 1993] system. GABIL is a concept learning system. The bit strings in GABIL represent a disjunctive set of propositional rules. GABIL has been used for both artificial and real world problems (breast cancer diagnosis) and was found to be roughly comparable in performance to C4.5, ID5R and AQ14. Extension operators were added to GABIL which are analogous to generalising the constraining of an spe-

cific attribute, or removing any constraint on a specific attribute. The addition of these operators improved the performance of the GABIL system. An introduction to GAs can be found in [Goldberg 1989].

GAs can be used in conjunction with other ML techniques. An examples of this is their use to help with model selection for SVMs as shown in [Chen, Wang & Lee 2004], or the use of a non generational GA, in this case a multi-start hill climber, in [Bongard & Lipson 2005] to improve the classification accuracy of a system which performs grammatical inference. GAs can be useful in cases where the selection of an appropriate ML technique is determined by complex factors of the problem such as in global surrogate modelling [Gorissen et al 2009]. Temporal Difference, TD, methods, [Sutton 1988], are often used to help with reinforcement learning problems. One issue with this is the choice of a function approximator to represent the value function. [Whiteson & Stone 2006] explore using an evolutionary function approximation, to automatically select function approximator representations that enable efficient learning.

Genetic Programming, GP, is similar in its evolutionary approach to GAs. However, where GAs usually represent the hypothesis as a bit string in GPs they are represented as computer programs [Koza 1992]. In a typical GP the program is represented as a parse tree. A function call would be represented by the root node of a tree and its arguments would be the descendants of the root node. Figure 2.14 is a representation of $\cos^2 x - \sqrt{x+y}$.

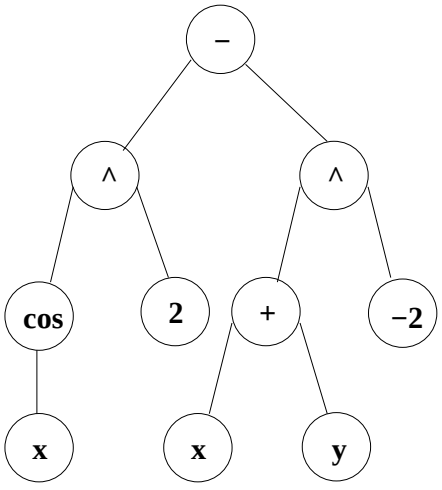


Figure 2.14: Parse Tree Representation of a Program in Genetic Programming

In GP the crossover operators typically perform some swapping of subtrees between programs. Mutation operators are more varied and changing operators, attributes and replacing subtrees with attributes are all possible. A comparison of various GP methods is given in

[O'Reilly & Oppacher 1994]. GP has been shown to be effective for some complex [Koza et al 1996] as well as simple problems. GP can be used in conjunction with other ML techniques, [Downing 2001] introduce a system, Reinforced Genetic Programming, which combines tree-based GP with reinforcement learning.

GAs and GPs both perform a randomised beam search for a maximally fit hypothesis. The nature of this search is quite different from that employed by most ML techniques. The use of crossover and mutation operators can result in radical differences between a hypothesis and its successor. Whereas in most ML techniques, like BACKPROPAGATION and the EM algorithm, changes are gradual tending towards the maximally fit hypothesis. [Teller & Veloso 2000] show how it's possible to develop a locally optimal update procedure for GPs so that they do tend towards gradual improvements leading to an optimal solution.

GAs and GPs are biologically inspired and the crossover and mutation operations on which they depend had little formal support. [Doerr Happ & Klein 2008] have shown that crossover operations can, in the case of the all-pair shortest-path problem, be proven to speedup the search for an optimal solution. It is possible for GAs and GPs to use forms of evolution which do not necessarily match those found in biology. In particular Baldwinian evolution [Baldwin 1896] and Lamarckian[13] inheritance [Lamarck 1809]. [Giraud-Carrier 2000] shows how Baldwinian evolution and Lamarckian inheritance can be used to improve learning for an ANN.

In their basic form neither GAs nor GPs use much domain knowledge. Like most ML techniques there are a number of ways in which domain knowledge can be made explicit in the learning process. The rules for crossover, mutation and selection can be modified to take account of domain knowledge, and the data can be modified to include domain knowledge. Similarly while there is no specific support for causality or causal discovery in the methodology of GAs and GP, it would be possible to add support by modifying the rules for crossover, mutation and selection.

2.13 Feature Subset Selection

In general ML systems perform best when given data containing only those attributes which are directly relevant to the system under study. The performance of many ML systems tend to degrade, in the time required for and accuracy of their analysis [Thrun et al 1991, John 1997],

[13] Although attributed to Jean-Baptiste Lamarck the ideas predate him and appear in [Darwin 1794].

and in memory requirements [Aha 1992], when supplied with additional unrelated data. In some cases, such as with naïve Bayesian, learners, see section 2.9, even the addition of relevant correlated features can adversely effect performance [Langley et al 1992]. It is reasonable to assume that it is not initially known precisely which attributes are relevant to a given system. Relevant domain information will suggest those attributes which may be of interest. Thus some method is needed which removes attributes that prove not to be related to the learning task. This weeding out of irrelevant, or unhelpful, attributes is called Feature Subset Selection, FSS, a good introduction is given by Guyon and Elisseeff [Guyon and Elisseeff 2003]. There are two common general techniques used to perform FSS, *filters* and *wrappers*. In addition for problems using linear regression the *lasso* [Tibshirani 1994] and similar methods have gained popularity. Ensemble techniques, see section 2.17, have also been used to improve the selection of relevant features [Tuv et al 2009]. As well as improving the performance of learners, the selection of *relevant* features potentially adds to our knowledge of the situation under investigation, as we would expect relevant features to be causally related to the target feature.

2.13.1 Filters

Filters are a method of selecting useful attributes from a dataset by performing statistical tests on the dataset. The idea is that only those attributes which show some form of correlation to the feature of interest will be included in the selected set. There has been a lot of research into identifying statistical tests for interesting attribute subsets [Ben-Bassat 1982, Kittler 1978]. This type of statistical analysis is relatively computationally inexpensive to perform, and this has encouraged the development of a number of more sophisticated search methods including, finding minimal feature sets [Almuallim & Dietterich 1991], formal methods for attribute rejection [Koller & Sahami 1996] and, formal definitions of feature relevance [Bell and Wang 2000]. The downside of this type of FSS is that it fails to take any account of the bias or specific requirements of the learner to be used.

2.13.2 Wrappers

Wrappers perform FSS using the learning algorithm to select the best set of attributes. A wrapper will repeatedly choose a subset of the available attributes and using the learner evaluate how well it allows the target function to be learnt. This makes the best use of the available data as the best selected subset will automatically cater for any bias in or limitations of the learner. However, if

there are N attributes then there are 2^N possible subsets. Since the learner is used on each selected subset the computational cost of analysing all possible subsets, for any non-trivial dataset, would be prohibitive. There is currently no best method for selecting an optimum subset, but this is an active area of research [Kohavi & Sommerfield 1995, Heiler et al 2001, Caruana & Freitag 1994, Sánchez-Maroño et al 2005, Van Dijck and Van Hulle 2006, Richards et al 2005]. Apart from the problem of computational cost there is also an issue with overfitting as the optimal wrapper seems to be dependent on both the learner and data set involved [Kohavi and John 1997].

2.13.3 Lasso

Lasso methods are those which seek to reduce some of the coefficients in linear programming problems to 0 thus in effect reducing the dimensionality of the problem. [Tibshirani 1994] introduced the, least absolute shrinkage and selection operator, lasso, as an improvement on the ordinary least squares, OLS, and ridge reduction methods for solving linear regression problems. This idea has been combined with other selection methods such as forward stepwise regression [Weisberg 1980] and least angle regression, LARS, [Efron et al 2004] and extended to cover group selection [Yuan and Lin 2006, Roth and Fischer 2008]. Support Vector Machines, SVMs, section 2.10, are linear learners and so these methods have been applied to them.

2.14 Data Management

There are numerous ways in which raw data for learners can be adjusted in order to improve the performance of the learner. The data may or may not contain explicit references to the required classification and this needs to be taken into account in the learning process. Probably the most common manipulation is simply restricting the data, given to the learner, to those attributes believed to be optimal to the learning task. This is often called Feature Subset Selection, FSS, see section 2.13. A different approach is to modify the data to allow relationship discovery that otherwise would not be possible, either because some data was not present in the dataset, or because the learner would not be able to link features with the unmodified representation. Lastly it is possible to improve our understanding of the accuracy of a learner on a task by varying the selection of training and test data from within the dataset.

2.14.1 Supervised Learning

One critical element of the training data is whether it contains the target function label, or classification. If it does then using it for learning is said to be *supervised* learning. This is optimal for learning to classify unseen data particularly in cases when domain knowledge is scarce.

2.14.2 Unsupervised Learning

If the training data is not classified then the learning is said to be *unsupervised*. Unsupervised learning refers to learning where the training data is unlabelled, in this case the learner attempts to group the data into similar sets, this is sometimes called clustering. An important issue with unsupervised learning is that the learner has to have a strong underlying assumption of the type of model with which to organise the data. The choice of this model, usually guided by domain knowledge, is crucial.

2.14.3 Semi-Supervised Learning

It is often the case that unlabelled data is plentiful, but labeled data is not. In these cases it may be possible for some of the data to be labeled by a domain expert giving a mix of labeled and unlabeled data. Learning from a dataset with some labeled and, usually mostly, unlabelled data is called *semi-supervised* learning. [Zhu 2008][14] provides a fairly comprehensive survey of semi-supervised learning in current literature.

2.14.4 Active Learning

When unlabeled data is easily available, but labelling or classifying the data is expensive in some fashion, then having the learner choose which data to label can lead to improved learning performance for a given cost. When it is the learner which chooses most or all of the data to label, the learning process is said to be an *active* one. By its very nature active-learning introduces a sampling bias. [Dasgupta & Hsu 2008], looks at the bias and uses a pruned tree structure to help select good clusters and hence appropriate sample data points. Active learning is currently a dynamic area of research with many different types of learner. Settles [Settles 2009][15] provides a good overview of current research.

[14] This is an updated online document and 2008 reflects the last update of which we are aware.
[15] This is an updated online document and 2009 reflects the last update of which we are aware.

2.15 Data Manipulation

Many ML techniques do not directly support the inclusion of either domain knowledge or time sequence information. Some ML techniques have been modified to allow the direct inclusion of domain knowledge, but it is common, when domain knowledge is required, to modify the data presented to the learner so that it includes pseudo attributes which represent domain knowledge. Similarly while some ML techniques have been modified to directly include time sequence information, it is more common to flatten the dataset to create sets of pseudo variables which allow the discovery of time sequence relationships. An example of the flattening process is shown in table 2.2.

Original Time Sequence Data			
Time	Variable A	Variable B	Variable C
T0	A0	B0	C0
T1	A1	B1	C1
T2	A2	B2	C2

Flattened Time Sequence Data						
Time	Variable A	Variable B	Variable C	Variable A'	Variable B'	Variable C'
T0	A0	B0	C0	A1	B1	C1
T1	A1	B1	C1	A2	B2	C2

Table 2.2: Time Sequence Flattening

2.16 Accuracy Estimation

When a machine learning technique has built a model from a dataset it is reasonable to enquire as to the likely accuracy of the model in classifying unseen data. There are a number of methods of manipulating the dataset to provide an estimate of this accuracy. Holdout is a technique in which some of the data, usually around a third, is kept back from training. The learner is trained on the dataset less the holdout data and tested on the holdout data. Using the definitions from [Kohavi 1995] we get, let V be the space of unlabeled instances and Y be the set of labels, $X = V \times Y$ is the space of labeled instances and $D = \{x_1, x_2, \ldots, x_n\}$ be a dataset consisting of n labeled instances, where $x_i = \langle v_i \in V, y_i \in Y \rangle$. A classifier C maps an unlabeled instance $v \in V$ to a label $y \in Y$ and an inducer I maps a given dataset D to a classifier C. $I(D,v)$ denotes the label assigned to an unlabeled instance v, by the classifier built by inducer I on the dataset D, that is $I(D,v) \equiv (I(D))(v)$. The accuracy of a classifier C is the probability of correctly classifying a random instance. So, $acc = Pr(C(v) = y)$ for some randomly selected instance $\langle v,y \rangle \in X$. Let

the holdout dataset be D_h such that $D_h \subset D$ and $D_t = D \backslash D_h$. The holdout estimated accuracy is

$$acc_h = \frac{1}{h} \sum_{\langle v_i, y_i \rangle \in D_h} \delta\left(I\left(D_t, v_i\right), y_i\right) \tag{2.67}$$

where $\delta(i,j) = 1$ if $i = j$ and 0 otherwise. In random sampling the holdout method is repeated k times and the estimated accuracy is derived from the average of the runs. In k-fold cross-validation the dataset D is randomly split into k distinct subsets of approximately equal size. The inducer is trained and tested k times; each time $t \in \{1,2,\ldots k\}$it is trained on $D \backslash D_t$ and tested on D_t. The cross-validation accuracy estimate is the overall number of correct classifications divided by the number of instances in the dataset. Let $D_{(i)}$ be the test set that includes the instance $x_i = \langle v_i, y_i \rangle$, then the cross-validation estimation of accuracy is

$$acc_{cv} = \frac{1}{n} \sum_{\langle v_i, y_i \rangle \in D} \delta\left(I\left(D \backslash D_{(i)}, v_i\right), y_i\right) \tag{2.68}$$

Leave one out is cross-validation with $k = n$. Stratified cross-validation is when each of the folds are stratified so that they contain approximately the same proportion of labels as the original dataset. While no general best method of accuracy estimation exists Kohavi [Kohavi 1995] showed that 10-fold cross-validation was a good approximation on a number of real world datasets.

2.17 Ensemble Augmentation Techniques

Ensemble methods, sometimes called a committee machines, generate multiple instances of a classifier from a given data set and incorporate some kind of voting or averaging mechanism to determine the classification or value of a new data item. There are two common methods of generating the group of learners, Bagging and Boosting and a third less common method Randomisation. These methods generate learnt instances by selecting multiple subsets of the original dataset with which to train the learners.

2.17.1 Bagging

Bagging, or bootstrap aggregating, is an ensemble method for increasing the accuracy of learners that are known to be unstable with respect to their training data. That is learners, such as decision trees, which can build significantly different models with only small changes in their training

data. Bagging was introduced by Breiman [Breiman 1996], although bootstrapping as a statistical technique predates it [Efron & Tibshirani 1994]. Bootstrapping involves repeatedly sampling the training set at random with replacement to produce a new training set of the same size. This new set can contain multiple instances of the same training item and in general will only contain around 63.2% of the unique examples in the original training set. The learner is then trained using the new training set. This process is repeated until a stopping condition, such as a given number of learners is generated or a given accuracy is achieved. A new data point is tested by voting, or averaging, over all the learners created. Stable learners, like k-Nearest Neighbour, k-NN, get little or no benefit from bagging as they produce similar models when given similar data. Averaging over similar models produces a result similar to that of any of the individual models. However, while k-NN learners are stable with variation in data, they are not stable with variation in features. Domeniconi and Yan, [Domeniconi and Yan 2004], have shown that ensemble techniques can improve the performance of k-NN learners when it is the features rather than the data that is varied.

2.17.2 Boosting

The idea behind boosting techniques is usually credited to an unpublished paper by Kearns [Kearns 1988]. In this the question is posed: can a set of weak learners create a single strong learner? Schapire [Schapire 1990] showed that in a probably approximately correct, PAC[16], learning model strong and weak learners are effectively equivalent and provides a method for using a PAC learner to provide an arbitrarily high accuracy. Boosting generates a learner using all the training data. Each data point is given a weighting, if the point is misclassified by the current learner its weighting is increased, if it is correctly classified its weighting is decreased. If the learner can directly use the weights then it is used on the updated training data, if not a new training dataset is created by sampling the updated training data selecting points with a probability relative to their weight. The classification of a new point is determined by voting, or averaging, between all the learners. A fixed number of learners can be created or some stopping condition, such as an upper bound on classification errors, can be used. AdaBoost developed by Freund and Schapire [Freund and Schapire 1995] is one of the most popular boosting algorithms and has many variations [Schapire & Singer 1999, Freund and Schapire 1995, Schapire 1997,

[16]PAC is a framework for mathematical analysis of machine learning introduced by Valiant [Valiant 1984].

Freund 1999]. While boosting is a useful technique it is not always applicable and can produce poor results in some circumstances such as when the training data has noise in its classification [Long and Servedio 2008].

2.17.3 Randomisation

Another approach to creating an ensemble is to create a random variation of some internal decisions made by the learner itself. Deitterich and Kong introduced a simple modification to the decision tree algorithm C4.5, see section 2.3, in which the decision about which split to introduce at each internal node of the tree is randomised from the twenty best splits [Dietterich & Kong 1995]. Further work on randomisation [Dietterich 2000], has shown randomising to perform similarly to Bagging although both are outperformed by Boosting on data with little classification noise.

2.17.4 Static and Dynamic Ensembles

In the above explanations of ensembles all the learners involved are trained using all the data and it is how all of their outputs are used and or combined which determines the behaviour of the ensemble, the input test value does not in any way change the parts of the ensemble involved in providing an answer. This arrangement is called a static ensemble. An alternative to this type of arrangement was proposed by Jacobs and Hinton at the connectionist summer school in Pittsburgh in 1988 and published as [Jacobs et al 1991b]. The idea, presented in terms of neural networks, is that if a task is known to be separable into a number of sub-tasks, then a different learner, or group of learners, can be used for each sub-task with a gating network determining which is the appropriate learner or learners for each training case. Hampshire and Weibel, [Hampshire & Waibel 1989], show how this can be done in the case where the sub-tasks are initially known, and Jacobs et al, [Jacobs et al 1991a], show how this can be done when the division into sub-tasks is not initially known. This type of ensemble where the input data determines which of the learners outputs are involved with the use if a gating network is called a dynamic ensemble, or a mixture of experts, or a committee machine, as the actual structure changes. Figure 2.15 shows an example of a dynamic ensemble. The output of the mixture of experts, in this case expert refers to a trained learner, o_m, is taken from one of the outputs of the individual learners, $o_1, o_2,$ or o_3 with a probability P_1, P_2 or P_3 determined by the gating network with the same input as each of the learners. In this kind of set up the gating network is used while

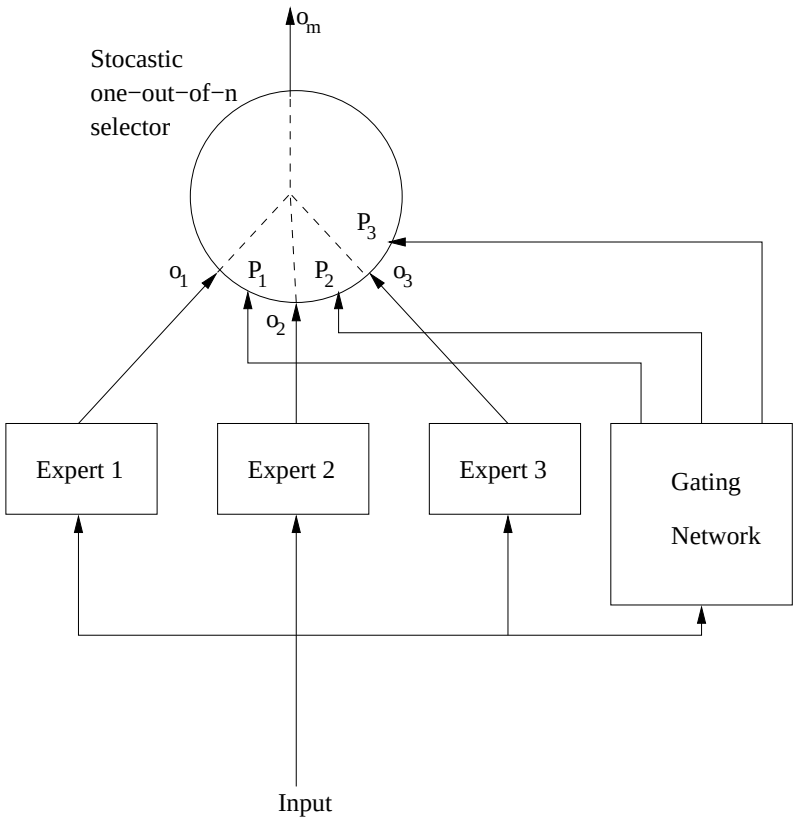


Figure 2.15: Dynamic Ensemble of Learners

training as well as for test data. This allows the feedback to the learners, in terms of weightings or other changes to effectively focus each learner on a different subset of the problem domain.

We have seen that a mixture of experts can effectively divide a problem domain into a number of sub-domains. ANNs have shown that it can be the case that features of the domain form a hierarchy. So, it should not be surprising that forming a hierarchical mixture of learners or experts can also prove to be a successful strategy to problem solving. This approach has been explored in a number of papers including [Jorden & Jacobs 1993, Bishop & Svensén 2003]. Figure 2.16 shows an example of this kind of ensemble. The output of the hierarchy, o_h is determined by a combination of the outputs of each of the separate mixtures of experts, o_{m1} and o_{m2} respectively, and a gating network all of which are fed the same input values.

2.17.5 Limitations of Ensemble Techniques

Ensemble techniques can significantly improve the accuracy of most learners. However, it is more difficult to interpret the model produced by the learner since multiple different models may exist. So, while prediction accuracy improves, understanding of the underlying model may not.

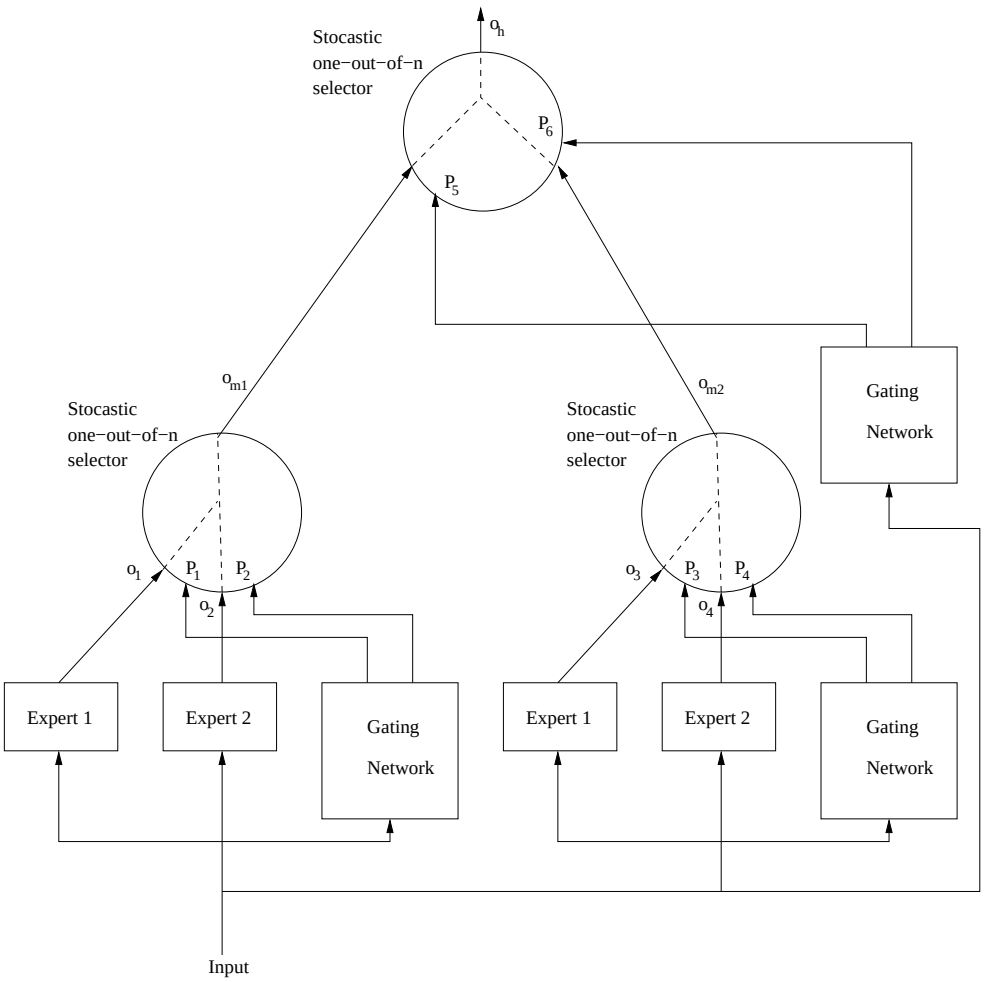


Figure 2.16: An Example of a Hierarchical Mixture of Experts

Some work has been done to increase the comprehensibility of boosted learners particularly with regards to rule-based learners [Fürnkranz & Widmer 1994, Cohen and Singer 1999].

2.18 Summary

In this chapter we started with a brief introduction to the history of ML. Then examine a number of different machine learners. It is clear that there are many different styles of machine learner which use a variety of techniques and varying amounts of domain information. Few current ML techniques make explicit any causal relationships in their hypotheses. BNs can be interpreted as causal relationships, and many other ML techniques are able, within the limitations of the learner's representation, to learn them. Most ML techniques allow the inclusion of at least some domain knowledge, and there is a general research effort to extend this ability. We have also seen that it is possible to manipulate the training data given to learners in a way that allows the inclusion of domain knowledge or to expose properties that otherwise could not be learnt.

A common research theme is the combining of ML techniques to make a single composite learner as potentially this allows the best use of the strengths of each of the different techniques. A variation on this is the use of ensemble techniques. This allows the combination of multiple copies of the same, or different, learners in a way that improves their overall performance.

ML is a very active field of research with continuing development of new learners and novel ways to maximise their capabilities. The range of learners examined do not display any particular strength in the area of causal discovery, this has resulted in the development of a number of specialised causal learning algorithms, a few of which we cover in more detail in chapter 4. Causal learners and other learners have a different focus, most standard learners are attempting to maximise their ability to predict values or classifications for new test data by maximising their use of whatever training data and information they are given. Causal learners are attempting to provide the most accurate representation of the causal relationships present in the data, using the data and other information they are given. This is important because the difficulty in understanding what has been learnt varies significantly between different learners. This is sometimes referred to as the difference between open-box and closed-box learners.

Open-box learners make what is learnt explicit, EBL and rules-based learners are examples of this type of learner. Closed-box learners tend to have no single item of knowledge you can point to as something learnt, instead they may have distributed knowledge, like the weights in

an ANN, or a general unspecified model, like the training data in a K-NN learner. So, all causal learners have to be open-box since what they produce is the model of causal relationships which best matches the data. It is also true that in general, with some notable exceptions like ILP, rules-based learning and EBL, the learners are able to make only limited use of domain information. This can be compared with LUMIN introduced in chapter 5 which is a dedicated causal learner that attempts to make maximal use of any available causal domain information.

Chapter 3

Causality, Causal Structure and Probability

In this chapter we move away from computer science and take a brief stroll through some philosophy. The following chapters of this thesis are devoted to discovering causal relationships. In this chapter we take a step back and try to examine causality itself. It seems odd to go in pursuit of something without being able to define the object of your pursuit. This chapter looks at attempts to define causality, and how they developed through time. We examine beliefs about the underlying mechanisms, if any, that give rise to causality. We explore, given a particular form of causality, what properties we might expect to see the system exhibit, and try understand how causal relationships can be learnt from data. We also have a very brief look at issues that arise with causal loops. This knowledge is required in order to develop a method of discovering causal relationships and to determine if the discovered relationship's properties match those expected of genuine causal relationships. The search for causal relationships and models of such relationships often involve both statistical and probabilistic reasoning. Statistics are used to determine the possible relationships and probability to select between them. It is curious then, that like causality while there is a ubiquitous 'common sense' understanding of probability, there is no certainty as to what probability is, or what a probability value means. While ideas about probability have developed over time, there are still numerous interpretations of probability being actively researched and supported. Some of the philosophical ideas which give rise to different interpretations of causality also give rise to different interpretations of probability. So, it is appropriate that we also investigate the meaning of probability. However, we start with a brief

examination of knowledge and belief as an understanding of these concepts is required to support arguments about learning and, as we will see, probability.

3.1 Knowledge and Belief

Philosophy involves the study of knowledge, belief and ultimately truth. It is important to understand what is required to turn a belief into knowledge as this is fundamental to the task of learning. In science observational and experimental evidence is used to suggest that a belief, or hypothesis, is true. However, despite the misnomer of scientific laws, modern science has no real concept of universal truths, rather it is the reliability of repeated measurement and predictive accuracy that act as a substitute for universal truths. All scientific *laws* are ultimately just beliefs which can be challenged by new, or more accurate, observations.

The philosophical study of the relationship between belief and knowledge goes back at least as far as Plato's dialogue, *Theætetus,* on the nature of knowledge. Following on from Plato it has generally been accepted that knowledge can be defined as "justified true belief". That is the relationship between belief and knowledge is that a belief is knowledge if the belief is true, the believer considers it to be true, and if the believer has a justification for believing it is true. While concise this relationship has been challenged by Gettier [Gettier 1963] and Dretske [Dretske 1981] leaving us in the situation where we know that

> knowledge is not, or is not merely, justified true belief
>
> – Fred I. Dretske.

This further raises the issue of what is a *true* statement and what constitutes *justification*? One line of argument is that a statement is true if it accurately reflects what it is describing. This is called the *correspondence* theory of truth [Allen 1993, Johnson 1992]. An alternative view is that a statement is true if it is consistent with a system of accepted statements. This is known as the *coherence* theory of truth. There are many versions of the coherence theory of truth Bradley [Bradley 1914], Putnam [Putnam 1981] and Young [Young 1987] present three differing versions. Another view is that truth is something that is learnt through experience and which additional experiences will not change. This led William James to define truth as the vanishing point toward which we imagine that all our temporary truths will some day converge. In some ways justification is simpler to understand, it seems fairly obvious that we are likely to feel a

statement is justified, only if it does not conflict with other statements we believe to be true. This type of reasoning lead Laurence BonJour to the conclusion that

> The only mode of justification for human knowledge is coherence.
>
> – Laurence BonJour

Machine learning can be thought of as a process which forms an initial hypothesis about some situation and attempts to improve the hypothesis until it is the best fit to the situation[1]. We always start with some knowledge and beliefs and attempt to add to our knowledge. In deductive learning coherence with our existing knowledge is required. Inductive learning is more focused on adjusting our hypothesis to correspond with the data. So, ML uses both correspondence and coherence, as measures of truth, for learning, although any given learner may use only one of these.

3.2 Defining Causality

What is causality? Causality is not easy to define, there is no one universally accepted definition of causality. The philosophical debate on the nature of causality is long running, ranging from at least Aristotle [Aristotle] then in Arabic and medieval philosophy under the label of occasionalism, through Francis Bacon [Bacon 1620], René Descartes [Descartes 1644], David Hume [Hume 1748], Immanuel Kant [Kant 1781], John Stuart Mill [Mill 1843], Bertrand Russell [Russell 1913], Karl Popper [Popper 1934], and Good [Good 1961] to the more current, David Lewis [Lewis 1973a], Sprites, Glymour and Scheines [Spirtes et al 1993], Judea Pearl [Pearl 2000], Nancy Cartwright [Cartwright 1989, Cartwright 1999, Cartwright 2002], and Jon Williamson [Williamson 2006]. Aristotle thought about causality in terms of a change in nature. He defines causality in terms of the four main agents of change. Considering these terms with reference to a statue we have:

material cause the bronze from which the statue is made

formal cause the form or shape of the statue

efficient cause the artisan who creates the statue

final cause to be an object of beauty

Bacon seems to have, in some ways, a modern view of causality

[1] A best fit for the situation is not usually a best fit to any given data set, see definition 3 section 2.2.

> Human knowledge and human power meet in one; for where the cause is not known the effect cannot be produced. Nature to be commanded must be obeyed; and that which in contemplation is as the cause is in operation as the rule.
>
> – Francis Bacon

However, he also appears to believe that not everything has a cause and that it can be folly to seek the causes of some things

> The like subtlety arises touching the infinite divisibility of lines, from the same inability of thought to stop. But this inability interferes more mischievously in the discovery of causes; for although the most general principles in nature ought to be held merely positive, as they are discovered, and cannot with truth be referred to a cause, nevertheless the human understanding being unable to rest still seeks something prior in the order of nature.
>
> – Francis Bacon

This contrasts with the modern scientific view for which the question 'Why?' or 'What causes this thing?', is always valid[2] even if the answer is frequently not known. Descartes retained many of the philosophical ideas of Aristotle, he believed God was "the efficient cause of all things". Although Descartes took God to be the primary efficient cause, he took a mechanical view of the secondary, or worldly, efficient causes. In particular he believed that there are fundamental rules or laws of nature and that natural phenomena can be explained by the mechanical application of these rules to the initial conditions. While ultimately controlled by the spiritual, this would seem to be, at least partly, a mechanistic view of causality. Descartes believed bodies interact by a transference mechanism: when x causes y a property of x is communicated to y. This he believed follows from the principle "Nothing comes from nothing".

> For if we admit that there is something in the effect that was not previously present in the cause, we shall also have to admit that this something was produced by nothing.
>
> – René Descartes

[2] Quantum mechanics could be interpreted to mean that some causes are unknowable.

Following on from Descartes, causal explanations were increasingly viewed as subsuming the events to be explained under general laws. Hume views cause and effect in terms of observed related events. So, that the observation that whenever event A occurs event B follows would serve as an example of cause and effect.

> A CAUSE is an object precedent and contiguous to another, and so united with it, that the idea of the one determines the mind to form the idea of the other, and the impression of the one to form a more lively idea of the other.
>
> – David Hume

However, Hume seems to consider the connections between cause and effect as being only due to observation, in essence, in the mind of the observer. Hence his assertion on the limitation of reasoning.

> From the first appearance of an object, we can never conjecture what effect will result from it. But were the power or energy of any cause discoverable by the mind, we could forsee the effect, even without experience; and might, at first, pronounce with certainty concerning it, by the mere dint of thought and reasoning.
>
> – David Hume

Hume also provided a second definition of causality

> We may define a cause to be an object followed by another, and where all the objects, similar to the first, are followed by objects similar to the second. Or, in other words, where, if the first object had not been, the second never had existed.
>
> – David Hume

this definition is in two parts, the first is a restatement of the regularity definition, but the second part introduces the idea of a counterfactual definition of causality. Hume separated causation from its earlier metaphysical associations allowing a secular definition, this paved the way for a more scientific approach to causation. Hume defined a number of "rules by which to judge of causes and effects". These included

> The same cause always produces the same effect, and the same effect never arises but from the same cause.
>
> – David Hume

which was pervasive in philosophical analysis of causality for some time. This is sometimes referred to as the regularity definition of causality. Hume did realise the weakness in using regularity as an indicator of causality in the lack of knowledge of the mechanism involved

> experience only teaches us, how one event constantly follows another; without instructing us in the secret connexion, which binds them together, and renders them inseparable.
>
> – David Hume

Kant disagreed with Hume's view and instead proposed that we could know things to be true even if we could not prove them. He came up with the idea of *synthetic a-priori* propositions. In this case synthetic means that they can be denied without contradiction, they do not contradict themselves or anything else that is true. So, while these propositions are not derived from experience, they can be used to help with analytic judgements. Thus while Kant sought to show that "everything that happens, that is, begins to be, presupposes something upon which it follows by rule," in his view the events are not "given", but are constructed by the organising activity of the mind. In this view causality is a mental rule imposed on events to facilitate organisation. In Kant's view the principal of causation is a synthetic *a-priori* one. John Stuart Mill extended the idea that related events form a causal relationship by saying that if an effect follows from a cause, the cause should not be taken to be a single factor, but rather all factors which are sufficient and necessary for the effect. Mill also introduced a deductive model for causal analysis

> An individual fact is said to be explained by pointing out its cause, that is, by stating the law or laws of causation of which its production is an instance.
>
> – John Stuart Mill

Mill also recognised that a number of causal laws might apply to a given situation and that the effects of each of those laws could be in opposition to each other. So, he concluded that an analysis should start by identifying and separating the laws relevant to the given situation

> The first mode, then, of explanation of Laws of Causation, is when the law of an effect is resolved into the various tendencies of which it is the result, together with the laws of those tendencies.
>
> – John Stuart Mill

Bertrand Russell considered causality from the viewpoint of how it effects science. He argues that many of the, then current, philosophical ideas on causality are either meaningless or incompatible with science. In particular he criticises the notion of "same cause, same effect" preferring to examine the relationships involved and concluding that while "sameness of relations" is better it is too simple and "sameness of differential relations" is the only equivalent correct phrase. Russell also contends that science does not assume a "law of causality", but rather the "uniformity of nature"

> The uniformity of nature does not assert the trivial principle "same cause, same effect," but the principle of the permanence of laws. That is to say, when a law exhibiting, e.g., an acceleration as a function of the configuration has been found to hold throughout the observable past, it is expected that it will continue to hold in the future, or that, if it does not itself hold, there is some other law, agreeing with the supposed law as regards the past, which will hold for the future.
>
> – Bertrand Russell

Popper's approach seems to have been to ignore any general rule(s) of causality. Indeed it appears that he sidestepped the issue of causality

> Of course, I have not solved the problem of how such interaction takes place; and indeed I suspect that this problem is insoluble - not only for interaction between mental and physical states, but more generally. For while, for example, we know that electrical charges repel one another, we have no 'ultimate explanation' of how they do it, even if we accept Maxwell's theory. We do not have any general theory of causality (at any rate not since the breakdown of Descartes' theory that all causality is push).
>
> – Karl Raimund Popper

choosing instead to concentrate on understanding how to evaluate a theory in light of the accuracy of its predictions, rather than consider any causal mechanisms involved. While causal understanding could produce perfect knowledge, Popper appears to concentrate on how to improve imperfect knowledge through hypothesis, deduction and observation. Good [Good 1961] avoided the use of counterfactuals when he tried to define a causal calculus. However there are issues with the inclusion of time in his definitions and a number of problems are still unresolved. Lewis, in [Lewis 1973a], attempts to define causation in terms of counterfactuals. He suggests causal claims can be explained in terms of counterfactual conditionals of the form "If A had not occurred, C would not have occurred". Various properties of this counterfactual causation are shown including transitivity and causal chains. Lewis' analysis uses possible world semantics. Its has been shown that there are problems with this definition of causality in terms of counterfactuals, particularly with regards to preempted causes. Lewis has since developed the idea [Lewis 2000], to counter some of the observed problems. Sprites, Glymour and Scheines, SGS, in their comprehensive book [Spirtes et al 1993] look at the statistical relationships that can be used to discover causal relationships in data, various algorithms which can explore these relationships, the reliability of such discoveries, and how to both optimally construct experiments to discover causal relationships and how to gain an understanding of the strengths and weaknesses of any discovered causal relationships. However, with all the analysis on the properties and methods of discovering and analysing causal relationships they avoid any definition of causality. Indeed they state

> We advocate no definition of causation ...
>
> – Sprites, Glymour and Scheines

it is instructive to see how much can be achieved by using some of the known, expected, likely or possible properties of causality even without defining it. Pearl mainly considers causal relationships of the form

$$x_i = f_i\left(pa_i, \varepsilon_i\right), i = 1, \ldots, n \tag{3.1}$$

where x is an effect and pa are the values of all those variables which are thought to be the immediate causes of x. ε is the error caused by factors not included in the pa values. A series of equations of the form 3.1 are a causal model if each equation represents a process by which the value of the variable x is determined. Equations of the form 3.1 can be replaced with the linear

equations

$$x_i = \sum_{k \neq i} \alpha_{ik} x_k + \varepsilon_i, \; i = 1, \ldots, n \tag{3.2}$$

This is the standard form of equations used in linear Structural Equation Modelling, SEM, [Wright 1921, Haavelmo 1943, Simon 1953]. Pearl conducts a comprehensive analysis of these causal relations in particular detailing many of the probabilistic relationships they entail. He also constructs a causal calculus based on intervention using the $do()$ operator which modifies a set of functions in the causal model, for example, to indicate the value of the variable X was set to be x you would write $do(X = x)$.

Definition 24. Pearl's **rules of induction** for this calculus are defined as follows: Let G be the DAG associated with a causal model, let $P(\cdot)$ stand for the probability distribution induced by that model. Let X, Y, Z, and W be disjoint subsets of variables (nodes) in G. We denote X and Y are independent given Z in G by $(X \perp\!\!\!\perp Y \mid Z)_G$. We denote the graph obtained from G by deleting all arrows pointing to or emerging from X by $G_{\overline{X}}$ and $G_{\underline{X}}$ respectively. The $\hat{}$ symbol identifies variables which are fixed in value by external intervention. We then have the following rules

1. Insertion/Deletion of Observations:

$$P(y \mid \hat{x}, z, w) = P(y \mid \hat{x}, w) \; if \; (Y \perp\!\!\!\perp Z \mid X, W)_{G_{\overline{X}}} \tag{3.3}$$

2. Action/Observation exchange

$$P(y \mid \hat{x}, \hat{z}, w) = P(y \mid \hat{x}, z, w) \; if \; (Y \perp\!\!\!\perp Z \mid X, W)_{G_{\overline{X}\underline{Z}}} \tag{3.4}$$

3. Insertion/Deletion of Actions

$$P(y \mid \hat{x}, \hat{z}, w) = P(y \mid \hat{x}, w) \; if \; (Y \perp\!\!\!\perp Z \mid X, W)_{G_{\overline{X},\overline{Z(W)}}} \tag{3.5}$$

where $Z(W)$ are the set of Z-nodes that are not ancestors of any W-nodes in $G_{\overline{X}}$.

Pearl makes extensive use of counterfactuals in his analysis and in that sense his work extends that of Lewis [Lewis 1973a]. Cartwright [Cartwright 1999] takes quite a different approach to causation, she does not believe that causal systems can reliably be decomposed. Similarly she

does not believe any one view of causality holds universally or indeed that any formal law would necessarily be universal. In [Cartwright 2002] she examines different methods of defining causal systems and finds that while each are useful in some circumstances none are universal. In fact she proposes

> I have presented the proposal that there are untold numbers of causal laws ...
>
> – Nancy Cartwright

which are related to

> different causal questions and different causal concepts with different criteria.
>
> – Nancy Cartwright

Cartwright seems to prefer a probabilistic view of causality, or at least its representation, to a structural one.

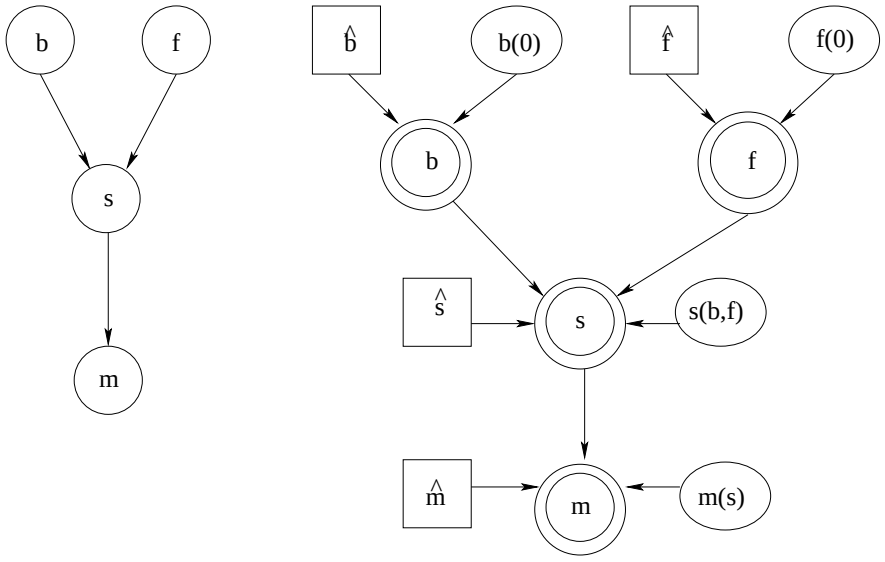


Figure 3.1: Causal Network and Corresponding Influence Diagram

A number of representations have been developed for causal relationships, each has its strengths and weaknesses. We will outline one here, taken from [Heckerman & Shachter 1994, Heckerman 1995a] as being both a reasonable representative of a graphical approach and one which explicitly uses deterministic functions. In this representation a causal network for a domain of chance variables, U, is defined as a directed acyclic graph where nodes represent the

variables in U, and each non-root node is the direct causal effect of its parents. A variable a is said to be *unresponsive* to another variable b if the value of a in any given situation will be the same regardless of the value of the variable b. *Set decision* is an intervention which changes the value of a variable without any other side effect, that is, the change only modifies the intended target variable and those other variables which are causally effected by the target variable, this is sometimes called an intervention. Suppose we have a collection of variables Y, which may contain both chance and decision variables, and a chance variable x. A *mapping variable* $x(Y)$ is a chance variable whose states correspond to all the possible mappings from Y to x. In this situation x will always be a deterministic function of the mapping variable $x(Y)$ and the variables Y. In general a mapping variable represents a counterfactual set of possible outcomes, only one of which we can observe. We can now state that a set of variables C are causes for x with respect to decisions D if $x \notin C$ and C is a minimal set of variables such that $x(C)$ is unresponsive to D. So, C is a cause of x with respect to D, if the way C affects x is not affected by D. In the case where C are the causes of x with respect to D we call the mapping variable $x(C)$ a causal mechanism. There are a number of other attempts to define causality in a way that is useful for computational analysis including [Pearl 1993, Pearl 1995, McCain & Turner 1997, Halpern & Pearl 2000] some of which explore what seem to us to be unexpected aspects of causal relationships [Pawlowski et al 2009]. Mostly these attempts make no effort to define causality, but rather assuming some unstated definition derive useful properties so the comment of [Freedman & Humphreys 1999]

> SGS do not give a reductive definition of 'A causes B' in non-causal terms. And their apparatus requires that you already understand what causes are. Indeed, the causal Markov condition and the faithfulness assumption boil down to this: causes can be represented by arrows when the data are faithful to the true causal graph that generates the data. Thus, causation is defined in terms of causation, with little value added. The mathematics in SGS will not be of much interest to philosophers seeking to clarify the meaning of causality.
>
> – Freedman & Humphreys

can be more generally applied than just to SGS's work [Spirtes et al 1993]. This work also adds little or nothing to the understanding of causality, we simply use what we perceive to be causal properties with no explanation of how such properties arise.

3.3 Foundations of Causality

There are alternative ways to view the underlying principles of causality. Once again the philosophical debate provides a selection of possibilities including, but not limited to, mechanistic, probabilistic, counterfactual, agency, and epistemic.

Mechanistic causality is about the physical processes which link the cause to the effect. Mechanistic views of causality often look at the system under investigation in terms of a conserved physical quantity like linear momentum, charge and mass-energy [Dowe 2000, Salmon 1998]. One problem with this approach is that it is not obvious how it would apply to causality in, say, economics. Some work has been done on this in terms of looking at a more general definition of the conserved quantity. Salmon argues, in [Salmon 1998], that the view of causality in economics is the same as the view of causality in physics and that economic causality is reducible to physical processes.

Probabilistic causality views causality in terms of probabilistic relationships between variables. It is not clear precisely what type of probabilistic relationships constitutes a causal relationship. The general intuition of probabilistic relationships is that of the Principle of Common Cause, introduced by Hans Reichenbach [Reichenback 1956].

> **Definition 25. Principle of Common Cause**: if two variables are probabilistically dependent then one causes the other or they are effects of common causes which screen off the dependence.

Thus two variables are probabilistically independent conditional on their common causes[3]. Patrick Suppes suggests causal relationships induce probabilistic dependencies. Thus screening off can be used to differentiate between variables that are common effect and variables that are cause and effect [Suppes 1970], although there are difficulties with this approach [Salmon 1998]. **Causal Dependence,** cause and direct effect are probabilistically dependent conditional on the effect's other direct causes, and the **Causal Markov Condition,** see definition 20 section 2.9.3, have been used by a number of people as a basis for defining probabilistic causal relationships [Pearl 1988, Spirtes et al 1993, Korb 1999, Pearl 2000]. Proponents of probabilistic causality often use Bayesian Networks as a representation of such probabilistic relationships.

Counterfactual causality represents causal relationships as subjunctive conditionals: C is a direct cause of E if and only if (1) if C were to occur then E would occur, or its chance of occurring

[3] A somewhat updated version of this principal can be found in [Penrose & Percival 1962].

would be significantly raised, and (2) if C were not to occur then E would not occur, or its chance of occurring would be significantly lowered. This is the approach explained by David Lewis [Lewis 1973]. Lewis' explanation of this idea draws on the idea of possible worlds[4], drawing conclusions about possible worlds close to our own. This type of approach seems very detached from a common sense type of reasoning exemplified by the Principle of Common Cause.

Agency causality [Price 1991, Menzies & Price 1993], represents causality from the point of view of an agent in trying to bring about effects by manipulating their causes. So in this view, C causes E if and only if bringing about C would be an effective way for an agent to bring about E. The strategy of bringing about C is deemed effective if a rational decision theory would prescribe it as a way of bringing about E. It is not clear to us in these accounts if causality exists outside of the agent, or if it is a secondary quality. Secondary qualities, like colour, while representing some physical property do not exist outside of the agent.

Epistemic causality, [Williamson 2004], is a mix of views on causality. It would seem to be an attempt to formalise the everyday notion of causality. The basis is that causality is a convenient tool in representing the world around us. That is, causality is a representation of our understanding of the world without necessarily being a physical property of the world. This idea is married with a limited form of objectivity based on background knowledge: If two agents with the same background knowledge disagree as to causal relationships then at least one of them must be wrong.

The view taken in this thesis is that causality is a fundamental property of the universe. We view the universe as deterministic, thus causal relationships must be deterministic. Causal relationships relate to a change in some quantity, the underlying principle is that if A is a cause of B then, with any other causes of B unchanged, a change in the value of A will entail a change in the value of B[5]. As with any change or measurement of change, time is a crucial factor, and allows additional constraints to be placed on causal relationships. Our view is that a cause must precede its effect(s) and that the minimum time lag between the cause and its effect(s) must be that required for information to travel, in space, from the cause to the effect, it is probably safe

[4]The Philosophy of possible worlds is originally attributed to Leibniz [Leibniz 1710], but the modern work is based more on the work of Lewis [Lewis 1986c] using semantics developed by Kripke [Kripke 1959, Kripke 1963].

[5]This is the basic principle behind the scientific use of experiments to determine and quantify causal relationships.

to assume that the maximum speed of this information exchange is the speed of light, c, in most cases. So we can define a causal function, f

$$y(t) = f(x_1(t), \ldots, x_n(t)) \tag{3.6}$$

where y is the effect and x_i are its direct causes and t is the time at which the values of y and the x_i are measured.

Direct causes are those which cannot be blocked although the effect could be altered by changes in other causes. Exposure to sunlight is not a direct cause of skin cancer since the application of sunblock, or increased skin pigmentation can nullify its effect. A direct cause of skin cancer is the interaction of ultraviolet light with live skin cells. If the amount of ultraviolet light reaching the live skin cells is measured then neither sunblock nor skin pigmentation are relevant. Ohm's law, $I = \frac{V}{R}$, relates the electrical current, I, flowing through a conductor to its resistance, R, and the voltage across it, V. While R and V are both direct causes of I they can be changed synchronously in such a fashion that I does not change, for example, if both V and R are doubled I remains unchanged.

Since the values in equation 3.6 are all time related and all should be measured at the same time this could be simplified to

$$y = f(x_1, \ldots, x_n) \tag{3.7}$$

with the understanding that the all the values are time dependent. If the minimum time for information to travel from a cause x_i to its effect y is t_{x_i} and $\triangle x_i(t)$ is the change in the value of x_i at a time t compared with its prior value at time $x_i(t-1)$ then we have

$$\triangle x_i(t) = x_i(t) - x_i(t-1) \tag{3.8}$$

and

$$\triangle x_i(t) \vDash \triangle y(t_{x_i y}) \tag{3.9}$$

where $t_{x_i y} \geq t + t_{x_i}$ and $a \vDash b$ is to be read as a entails b. This definition suggests that a causal relationship is a correlated one, but only in terms of altering a single causal variable while the others remain fixed. The simple case of Ohm's law showed that even with a very simple deterministic relationship its possible to alter multiple *causes* and to produce no change in their *effect*.

3.4 Causal Systems

We assume a mechanistic version of causality, where the causality is a reflection of underlying physical properties and processes. What does this mean in terms of what behaviour we would expect to see from a causal system? We would expect causal systems to be predictable, assuming no other changes, similar starting conditions should produce similar cause and effect chains[6]. The underlying physical processes are assumed to be invariant. So, the observed cause and effect relationships should also be invariant.

> We may regard the present state of the universe as the effect of its past and the cause of its future. An intellect which at a certain moment would know all forces that set nature in motion, and all positions of all items of which nature is composed, if this intellect were also vast enough to submit these data to analysis, it would embrace in a single formula the movements of the greatest bodies of the universe and those of the tiniest atom; for such an intellect nothing would be uncertain and the future just like the past would be present before its eyes.
>
> – Pierre Simon Laplace, A Philosophical Essay on Probabilities

Since physical processes underlie the cause and effect link, the cause must occur before the effect. Physical process take a finite amount of time, which would allow a determination of the minimum time that could occur between a cause and its effect. However, without knowing or making assumptions about the underlying mechanisms involved, all that can be determined is the minimum delay between a change in a cause and its effect, see equation 3.9.

Things don't just happen. "*It fell apart in my hands*", is possible, but whatever *it* is would have had a cause to fall apart at that moment. When changes occur there will be a reason for those changes. So, provided the underlying causes are observable, and at least sometimes observed, it may be possible to determine the relationships involved between effects and their causes. When some or all of the underlying processes are not observed, probabilistic relationships may still be observed. Consider a ball bouncing around the inside of an irregular sphere, which has some weak points that when struck by the ball would cause it to shatter. What would be seen is that at some point in time the sphere would shatter with no apparent cause. Now if we had thousands

[6] In some situations small changes to the initial conditions can lead to large changes over time, but the cause and effect chains are still similar. See [Wolfram 2002] chapter 7 for an introduction to chaos theory.

of identical spheres, we might notice that while we could not predict when any individual sphere would shatter, we could say how long it took for half the spheres to shatter. Since the probability of a ball striking a weak spot is determined by the ratio of the area of the sphere that is weak to that which is strong, a fixed value, we would expect the average half life of the spheres to be constant. In this instance a deterministic process which is unobservable gives rise to a statistical relationship.

With mechanistic processes as the underlying link between cause and effect, there is the possibility of finding the mechanism that links a cause to an effect. So, when a causal link is proposed, a search for an underlying mechanism can add to the confidence of the proposal. However, not finding a mechanism does not, by itself, invalidate the proposed causal link. With causal mechanisms as with many things

> Absence of evidence is not evidence of absence.
>
> – Carl Sagan

This is an important point as it has become fashionable to dismiss possible causal relationships where there is no *currently* known physical mechanism connecting cause to effect.

3.5 Learning Causality from Data

Even assuming a mechanistic view of causality it is in general not possible to directly observe causal relationships in observational data. However, causal relationships will usually lead to statistical relationships. Since a change in a cause usually leads to a change in an effect, we would expect there to be shared information between causes and their effects. So, statistical relationships can act as indicators of possible causal ones.

It is a common fallacy that statistical correlation is always an indicator of causal relationships, *cum hoc ergo propter hoc* (Latin for "with this, therefore because of this"). Alas the world is not so simple and it is clear that many cases of strong correlation do not imply a causal link. In some places skirt lengths and stock prices have been highly correlated, as stock prices go up, skirt lengths get shorter[7]. So, a method to differentiate causal from chance relationships is needed. Granger [Granger 1969] proposes a simple if somewhat impractical definition of causality.

[7] It is possible this is not coincidence, but its exploration is beyond the scope of this thesis.

Definition 26. Let X be a stationary stochastic process and $\overline{X}$ represent all its past values. Let $P(X \mid B)$ be the optimum, unbiased, least-squares predictor of X using the set of values B.

$$\text{If } \sigma^2(X \mid U) < \sigma^2\left(X \mid \overline{U-Y}\right) \tag{3.10}$$

we say that Y is **causing** X, where $\sigma^2(X)$ is the variance of the predictive error series derived from using the optimum, unbiased, least-squares predictor of X , U is all the information in the universe and $\overline{U-Y}$ is all this information apart from the specified series Y.

This simply boils down to if we can make a better prediction of the value of X by knowing the value of Y than without it, we can claim that Y is a cause of X. Alas we do not know U and it is not possible to say that the data available is a reasonable approximation of U unless we already know the underlying causal relationships involved. There have been a number of algorithms developed which attempt to detect causal relationships in data. Some of these will be examined in chapter 4 and a new one introduced in chapter 5.

Demonstrating that a particular relationships is causal is not easy. The simplest approach is to determine experiments where controlled changes can be made to assumed causes and then measurements taken of the impact of this on their assumed effects. This is the normal scientific experimental method. However, in the case of observational data or where a controlled experiment is not possible or ethical, a different approach is required. One option is to use the verifiability principle.

Definition 27. Verifiability Principle: The statement is literally meaningful, it expresses a proposition, if and only if it is either analytic or empirically verifiable.

Since the relationship was suggested by observational data it should therefore be empirically verifiable, at least for the dataset used in the analysis. Certainty in causal relationships is rare even when seemingly good experiments are available. Consider the situation of a sealed box which has a meter indicating positive or negative changes, and a dial which may be turned to the left or right. Suppose that on each occasion the dial is turned to the left the meter shows a negative change, and on each occasion the dial is turned to the right the meter shows a positive change. Even in this most clear cut case we cannot conclude that there is a causal relationship between the dial and the meter, and certainly not determine the details of any actual causal relationship. We can point to a likely relationship, but without knowledge of the contents of the box and the

actual physical relationship between the dial and the meter, if any, we cannot know the details of any relationship which exists.

A causal relationship suggested by a single dataset should be naturally suspect. Evidence suggesting a physical mechanism for the relationship will help to increase confidence in the reality of the relationship. Confirmation of the apparent relationship in different datasets also helps to confirm its reality. This conforms to Popper's ideas on hypothetico-deductive discovery [Popper 1934]. In this situation we have a dataset which through inductive means suggests a causal relationship. This relationship will then have consequences which should be detectable in other datasets. Thus the proposed causal relationship should be falsifiable.

The simple definition of a causal relationship given in equation 3.7 suggests that there should be a statistical relationship between a change in a cause and a change in its effect(s). Since we are talking about changes over time, and with a nod to Russell, given a change in x_i we have

$$\begin{aligned} \frac{dy}{dt} &= \frac{df(x_1,\ldots,x_n)}{dx_i} \cdot \frac{dx_i}{dt} \\ &= g(x_1,\ldots,x_n)\frac{dx_i}{dt} \end{aligned} \tag{3.11}$$

where g is a function of $x_1,\ldots,x_n$ since each of the x_i are independent. However, as has been shown by Cartwright and by Neufeld [Neufeld & Kristtorn 2005] and in the Ohm's law example above, it is possible to have changes in multiple causes independent of changes in their effect(s). While this is possible it generally requires quite contrived examples and we can argue that is it usually safe to assume that a lack of shared information, or correlation, between two variables implies no causal link between them. A small amount of data which shows no link where one is expected is suspect because any results from a small amount of data should be suspect. A significant body of data that shows no link where one is expected suggests that the expectation may be wrong. A causal relationship is only certain when the physical processes involved in the relationship are understood.

3.6 Causal Loops

Causal loops occur when there is some feedback from an effect to its cause. That is, suppose it is known that an attribute, A, is a cause of another attribute, B. It is then discovered that B is, directly or otherwise, a cause of A. In terms of the symbols used in definition 26 above if we

have

$$\sigma^2\left(A \mid \overline{U}\right) < \sigma^2\left(A \mid \overline{U - B}\right) \tag{3.12}$$

and

$$\sigma^2\left(B \mid U\right) < \sigma^2\left(B \mid \overline{U - A}\right) \tag{3.13}$$

then we have what Granger calls feedback and we term a causal loop. Since we are assuming mechanistic causality, unless A and B are properties of the same object, there will be some time delay between a change in an attribute and the propagation of that change to the other attributes it effects. A simple example of this is, when our house is cold we turn the air-con off and the heating on, when its hot we turn the air-con on and the heating off. So the temperature of the house effects the state of both the heating and the air-con, through us as an intermediary. In turn the heating and the air-con effect the temperature of the house.

With causal loops time separation in the data is critical. In general we need to assume that in a given data record the values of the attributes reflect a time period short enough for any feedback not to have occurred. If the values of the attributes in a record refer to different times sufficiently separated so that feedback may have occurred, it may not be possible to determine simple causal relationships, and it is unlikely that causal loops can be distinguished.

3.7 Logical Implication, Entailment, Causality and Bayes Rule

It is of interest to think about the relationship between logical implication, causal relationships and Bayesian analysis.

> **Definition 28. Logical implication**, also called the material conditional, is a relationship between two logical propositions, A and B commonly indicated by the symbol $\rightarrow$, $A \rightarrow B$ indicates that A implies B. The truth value of this expression is given by table 3.1.

A	B	$A \rightarrow B$
F	F	T
F	T	T
T	F	F
T	T	T

Table 3.1: Logical Implication Truth Table

The name and 'common sense' interpretation of implication might suggest its use as an indicator of causal relationships, that is, if A implies B we could imagine that A is a cause of B.

Although $A \rightarrow B$ is equivalent to the statement "`If` A `then` B", in normal use such a statement suggests a causal relationship between A and B. So, while the statement "`If` (U) all bachelors are unmarried `then` (C) the speed of light in a vacuum is constant" is logically true, $U \rightarrow C$ is true, in normal use the sentence would be taken to be false or meaningless since there is no known link between bachelors and the speed of light in a vacuum. Also, as table 3.1 shows logical implication tells us nothing about the consequent proposition, B, if the antecedent proposition, A, is false. This leads to the statement "`If` 2 is odd `then` 2 is even" being logically true, but confusing, meaningless or even false in a natural language sense. However, as Strawson commented [Strawson 1950]

> Neither Aristotelian nor Russelian rules give the exact logic of any expressions in ordinary language, for ordinary language has no exact logic
>
> – Peter Frederick Strawson

The desire to have some general relationship between the antecedent and consequent elements of the implication relationship has lead to a branch of logic called relevance logic. This form of logic is one in which implication has a meaning which is closer to our every day understanding. Early work on a logical framework to include relevance was shown in Lewis and Langford's work, Symbolic Logic [Lewis & Langford 1932], and its applicability to causal logic has also been investigated in, for example, [Henderson 1954]. The major modern foundation for relevance logic, however, is Anderson and Belnap's, Entailment: The Logic of Relevance and Necessity [Anderson & Belnap 1975, Anderson, Belnap & Dunn 1992].

The entailment operator, commonly indicated with the $\models$ symbol, $A \models B$ indicates that A entails B, is an operator which provides a logical relationship between statements.

> **Definition 29.** A common form of the definition of **entailment** is:
> A statement A entails a statement B if and only if A's being true is a sufficient condition for B's being true, and B's being false is a sufficient condition for A's being false[8].

It would seem that entailment might be a more suitable operator to indicate a causal relationship, it certainly ensures a close relationship between the logical sentences. However, this relationship is such that it precludes non-monotonic and defeasible reasoning and as such requires that we

[8] A being not true might be a more generally useful definition, see [Ginsberg 1972, Facione 1977].

know all of the entailing factors of a causal relationship, making it generally unsuitable as a model while searching for causal relationships.

Bayes Theorem provides a method for assessing the most likely relationships in a system when given information about the behaviour of the system. Equation 2.42 tells us that to find the most likely hypothesis h about the relationships in a system, we need to find the hypothesis which maximises the likelihood of obtaining the data D which was produced by the system. So, in searching for the most likely hypothesis we are looking for the one which maximises the value of $P(D \mid h)$. The expression $P(D \mid h)$ suggests a causal relationship between D and h, which is after all the target of the search. This could be interpreted as implication with a level of support, that is, $P(D \mid h)$ could be interpreted to mean that $h \rightarrow D$ with a level of support equal to $P(D \mid h)$, the obvious interpretation of the range of values $[0,1]$ would be that 0 indicates no support and 1 indicates certainty. This could be seen as interpreting probability in a logical framework as explored by Carnap [Carnap 1950], see section 3.8.2. However, there is no guarantee that our best guess hypothesis is even close to the actual relationships in the system, regardless of the value of $P(D \mid h)$. In general it is not safe to assume that the existence of a hypothesis which gives a large value for $P(D \mid h)$ indicates a causal link between h and D. This is just another instance of the saying 'correlation does not imply causation', although we will show in chapters 4 and 5 that it can be used as an indicator of where to search for possible causal relationships.

A potentially more troubling issue when considering using Bayes' rule in the context of causal relationships is that of the different symmetries of probabilistic and causal relationships. In terms of probability calculus there is no difficulty in having $P(D \mid h)$ and $P(h \mid D)$, they are both valid, meaningful expressions. However, if we interpret $P(D \mid h)$ to mean that h is a cause of D, then $P(h \mid D)$ would suggest that the effect is driving its cause. Humphreys [Humphreys 1985] examines this issue in relationship to propensities, see section 3.8.4, but some of that argument is relevant to the potential problems in using conditional probabilities to represent causal relationships.

3.8 Interpreting Probability

Although probability is used extensively in the field of machine learning, the meaning of probability values and determining what they should be, is something which can be difficult to define. There are a number of possible interpretations of the meaning of probability. Most interpreta-

tions of probability conform to the axioms set out in Kolmogorov's, Foundations of the Theory of Probability [Kolmogorov 1933].

Definition 30. Kolmogorov's Probability Axioms can be stated as:

Let Ω be a non empty set of all possible outcomes. A field on Ω is a set F of subsets of Ω that has Ω as a member, and that is closed under complementation and union. Let P be a function from F to the real numbers obeying:

Non-negativity: $P(A) \geq 0$, for all $A \in F$.

Normalisation: $P(\Omega) := 1$.

Finite additivity: $P(A \cup B) := P(A) + P(B)$ for all $A, B \in F$ such that $A \cap B = \varnothing$.

We call P a probability function and (Ω, F, P) a probability space.

It is not uncommon, although it is not a point on which there is universal agreement, to strengthen the closure assumption on F, requiring it to be closed under complementation and countable union. Under these conditions F is called a sigma field on Ω.

Definition 31. Countable additivity: If $A1, A2, A3\ldots$ is a countably infinite sequence of, pairwise, disjoint sets, each of which is an element of F, then we have

$$P\left(\bigcup_{n=1}^{\infty} A_n\right) := \sum_{n=1}^{\infty} P(A_n) \tag{3.14}$$

Kolmogorov states that while using only models with countable additivity is an arbitrary restriction is does allow idealised models of real random processes. This axiom also allows the integration of probability theory with measure theory[9]. While Kolmogorov's axioms are commonly those used they are neither the only possible axioms of probability nor the only ones which have been explored and found to be useful. However, examination of alternative probability frameworks is beyond the scope of this thesis.

Definition 32. Conditional probability of A given B, written as $P(A \mid B)$, can be defined as:

$$P(A \mid B) := \frac{P(A \cap B)}{P(B)} \text{ where } P(B) > 0 \tag{3.15}$$

Some formulations of probability define conditional probability as a primitive notion, and provide an axiom for it directly[10].

[9] See [Halmos 1974, Bartle 1995, Dudley 2002] for more details on measure theory and probability.

[10] See for example [Popper 1934, Spohn 1986, Roeper & Leblanc 1999].

3.8.1 Classical Probability

The classical interpretation of probabilities is that explained by Laplace [Laplace 1814]. This interpretation of probability fits well with the games of chance, of the day, which were likely the inspiration for it. Laplace explains the concept as follows (translated for the 1951 English edition):

> The theory of chance consists in reducing all the events of the same kind to a certain number of cases equally possible, that is to say, to such as we may be equally undecided about in regard to their existence, and in determining the number of cases favorable to the event whose probability is sought. The ratio of this number to that of all the cases possible is the measure of this probability, which is thus simply a fraction whose numerator is the number of favorable cases and whose denominator is the number of all the cases possible.
>
> – Pierre-Simon Laplace

This allows the assignment of probabilities in the absence of any evidence and in the presence of symmetrically balanced evidence. There are difficulties with this explanation including the issue of what events are of "the same kind", the role of the undecided entity in "we may be equally undecided" and in its applicability to infinite probability spaces and to irrational probability values, such as those proposed by quantum mechanics. Laplace also assumes the 'principle of indifference'.

> **Definition 33.** The **Principle of Indifference** states that whenever there is no evidence favouring one possibility over another, they have the same probability.

Although this idea had been around for some time, it was Keynes [Keynes 1921], who coined the phrase. Jaynes [Jaynes 1968], showed that the classical interpretation can be extended to cover countably infinite probability spaces by generalising the principle of indifference to that of maximum entropy.

> **Definition 34.** The entropy of a discrete distribution is given by equation 2.2. The **Principle of Maximum Entropy** requires that we select from the family of all distributions consistent with our background knowledge the distribution that maximises this quantity.

For infinite distributions this usually requires the imposition of additional constraints, there is currently no theory of how this can be done in general. A further challenge to classical probability was posed by Bertrand [Bertrand 1889], in the form of a probability paradox[11]. A solution was proposed by Jaynes [Jaynes 1973], based on the principle of 'maximum ignorance'.

> **Definition 35.** The **Principle of Maximum Ignorance** states that we should not use any information that is not given in the statement of the problem.

Basic classical probability allows no means of altering probabilities in light of new evidence. Laplace extended the basic theory with the Rule of Succession.

> Thus we find that an event having occurred successively any number of times, the probability that it will happen again the next time is equal to this number increased by unity divided by the same number, increased by two units.
>
> – Pierre-Simon Laplace

> **Definition 36.** The **Rule of Succession** links probabilities to the frequency of observed events, it can be formulated as:
>
> $$P(\text{success on } N+1\text{st trial} \mid N \text{ consecutive successes}) := \frac{N+1}{N+2} \tag{3.16}$$
>
> where 'success' is simply the occurrence of a particular event, such as a coin toss resulting in heads.

Classical probability's reliance on the principle of indifference can be seen as a problem as it appears to provide probability values in a situation where there is ignorance of those very same values [Fine 1973].

> If we are truly ignorant about a set of alternatives, then we are also ignorant about combinations of alternatives and about subdivisions of alternatives. However, the principle of indifference when applied to alternatives, or their combinations, or their subdivisions, yields different probability assignments
>
> –Terrence L. Fine

[11] A good explanation of the form of this paradox, and the subject in general, can be found in [Hájek 2010].

This is often relevant in the context of Bayesian learning where it is not uncommon to use the principal of indifference in assigning prior probabilities where no better estimates exist[12]. However, in a situation of total ignorance it is uncertain that any specific value or interval, other than the interval $[0,1]$, can be supported as even a prior probability value without additional information.

3.8.2 Logical Probability

Logical theories of probabilities also use the idea that probabilities can be assigned by examination of all the possible outcomes in a given situation. They extend classical theories by allowing for differing probabilities and allow the computation of probabilities based on evidence even when it is not symmetrically balanced. Logical interpretation of probability seeks to determine the degree of support that a piece of evidence E confers upon a given hypothesis H this can be written as $c(H,E)$. Carnap [Carnap 1950] explored logical interpretations of probability and devised a class of very simple languages consisting of a finite number of logically independent monadic predicates applied to countably many individual constants or variables, and the usual logical connectives. The strongest consistent statements that can be made in a given language are called *state descriptions*. This use of a confirmation function, $c(-,-)$, can be viewed as generalising deductive logic and its idea of implication, to inductive logic with the notion of 'degree of implication'.

> **Definition 37.** Any probability measure $m(-)$ over the state descriptions extends to a measure over all sentences and induces a **confirmation function** c:

$$c(H,E) := \frac{m(H\&E)}{m(E)} \tag{3.17}$$

While there are an infinite number of possible probability measures, giving rise to infinitely many possible confirmation functions, Carnap supported a particular measure, $m*$. A *structure description* is a maximal set of state descriptions, each of which can be obtained from another by permutation of the individual names. $m*$ assigns each structure description an equal measure, which in turn is divided equally among their constituent state descriptions. The confirmation function can be generalised to a continuum of functions $c\lambda$. A *family* of predicates is a set of predicates such that, for each individual, exactly one member of the set applies. Carnap

[12] In Bayesian Networks this if often called using uninformative or uniform priors.

[Carnap 1963] further developed this based on a language containing only one-place predicates. Adding a number of axioms concerning the confirmation function c, symmetry and inductive learning he found that they imply for a family $\{P_n\}, n = 1, \ldots, k\,(k > 2)$:

$$c\lambda \text{ (individual s} + 1 \text{ is } P_j, s_j \text{ of the first s individuals are } P_j) := \frac{(s_j + \lambda/k)}{s + \lambda} \tag{3.18}$$

where λ is a positive real number. λ varies the relative weight given to evidence and *a-priori* probabilities, when $\lambda = 0$ the conditional probabilities are the same as the relative frequencies, when $\lambda = \infty$ evidence is ignored and the probabilities used are only the *a-priori* ones.

There are a number of possible problems with this type of probability interpretation in particular to do with inductive logic, what value of λ to use and how it should be chosen, and the syntactic nature of the learner. Goodman [Goodman 1955] showed with his "new riddle of induction" that inductive logic must be sensitive to the meanings of predicates, thus it appears that Carnap's approach is likely to fail, but this is an area of active debate[13].

3.8.3 Frequency Interpretations

It has been recognised for a long time that relative frequencies have a strong relationship to probabilities. Frequency interpretations suggest that this relationship is identity, that is, in the case of a finite reference class, such as tossing a coin 100 times, we have the probability of an attribute A, say heads, in a finite reference class B, the 100 coin tosses, is the relative frequency of actual occurrences of A within B. This differs from the classical interpretation which counts all the possible outcomes of a given experiment, instead this finite frequentism counts actual outcomes. This makes it compatible with an empiricist world view. Finite frequentism was largely developed by Venn [Venn 1866], who in his discussion of the proportion of births of males and females, concludes:

> probability **is** nothing but that proportion
>
> – John Venn

Finite frequentism gives an operational definition of probability, unfortunately is it easy to see how the essence of probability itself makes this kind of definition of very limited use. If a coin is

[13] Patrick Maher and others argue that Carnap's posthumous publications 'A Basic System of Inductive Logic, Parts I & II' overcome many of the objections to his earlier work.

tossed once and lands heads, then this form of frequentism would assign the probability of 1 to any future toss of the coin landing heads. The general issue of a single case is common enough to be given its own name, the 'problem of the single case'. Similarly if a coin that is known to be fair[14], is tossed 10 times and lands heads 9 times, it would yield a probability of it landing heads in a future toss as 0.9. The basic problem here is that a finite reference class of size n can only produce relative frequencies at a certain level of accuracy $1/n$. As with the previous interpretations of probability finite frequentism rules out the possibility of irrational probabilities which are required by some current physical theories like quantum mechanics. Another problem with finite frequentism is that the intermediate probabilities generated by any grouping of objects and attributes is as likely to be simply a tally as it is a genuine probability. Partly in response to these problems a number of frequentists investigated infinite reference classes [Venn 1866, Reichenbach 1949, von Mises 1957].

An infinite reference class allows probabilities to be identified with limiting relative frequencies of events or attributes. One problem here is that for the overwhelming majority of possible reference classes there is no available infinite sequence of trials. So, now a probability has to be associated with a hypothetical or counterfactual limiting relative frequency. Where a finite sequence of trials exists we need to construct a hypothetical infinite extension, the probabilities are then what the limiting relative frequencies would be if the sequence were so extended. This is obviously no longer a match for an empiricist world view.

A problem arises when considering extending finite sequences to infinite sequences with the intention of looking for a limiting relative frequency, the order of the elements in the sequence can be significant. Suppose the series we are investigating is that of positive integers, and the initial sequence we have is: $1,3,5,2,7,9,11,4$. If we continue this sequence and ask what is the probability that a given number is even, the answer we get is $1/4$ not $1/2$ as we would expect. While in this constructed case it may be easy to say what the 'natural' ordering of the sequence should be, in general there is no reason to believe we would know the appropriate ordering, or alternatively know that we were using an inappropriate one, or even if such a special ordering exists. All this leads to the question, why should one ordering be privileged over others?

A well known problem for any version of frequentism is that relative frequencies must be relativised to a reference class. Suppose that you are interested in the probability that you will live to age eighty. Which is the appropriate reference class? The class of all people? All people

[14] In this case it would probably mean it was known to be of a balanced construction.

of your gender? All people of your body mass index? This is an example of what is usually called the reference class problem. As previously noted the problem is only compounded for limiting relative frequencies, as probabilities must be relativised not merely to a reference class, but to a sequence within the reference class. This might be called the *reference sequence problem*.

A partial solution to this problem is to limit the type of sequences which can be used. Von Mises [von Mises 1957] restricts the sequences to what he calls collectives. Collectives are hypothetical infinite sequences of attributes of specified experiments that meet certain requirements. Call a place-selection an effectively specifiable method of selecting indices of members of the sequence, such that the selection or not of the index i depends at most on the first $i-1$ attributes. He then defines two axioms:

> **Definition 38. Axiom of Convergence**: the limiting relative frequency of any attribute exists.

> **Definition 39. Axiom of Randomness**: the limiting relative frequency of each attribute in a collective ω is the same in any infinite sub-sequence of ω which is determined by a place selection.

Church [Church 1940] supplies a precise notion of a place selection as a recursive function. The problem of an infinite number of possible reference sequences, which may generate differing probabilities, means that from an infinite frequentist point of view a probability only applies with respect to a particular collection. Von Mises regarded the problem of the single case as meaningless:

> We can say nothing about the probability of death of an individual even if we know his condition of life and health in detail. The phrase 'probability of death', when it refers to a single person, has no meaning at all for us
>
> – Richard von Mises

However, it is not clear using this type of reasoning that a finite collection of any size would be considered suitable for the assignment of probabilities. Von Mises could have found himself in the position of not being able to say much about the 'probability of death' even when it refers to one of six thousand million individuals.

3.8.4 Propensity Interpretations

Propensity interpretations of probability, [Popper 1957, Peirce 1958, Popper 1959], treat it as a physical propensity, a tendency of a given type of physical situation to yield an outcome of a certain kind, or to yield a long run relative frequency of such an outcome. A propensity interpretation of probability solves the problem of the single case. Popper [Popper 1957] explains his propensity theory with reference to its ability to encompass quantum mechanical probabilities. Popper further advanced his theory in [Popper 1959]. According to this interpretation, a probability p of an outcome of a certain type is a propensity of a repeatable experiment to produce outcomes of that type with limiting relative frequency p. The use of limiting relative frequency might seem to lead to von Mises' style of frequentism. An alternative interpretation by Giere [Giere 1973], explicitly allows single-case propensities, with no mention of frequencies. Under Giere's interpretation a probability is just a propensity of a repeatable experimental set-up to produce sequences of outcomes. This leaves a potential problem in defining the relationship between probabilities and frequencies.

What a propensity is varies depending on the theory being examined. Popper's theory for a fair die is that it has a propensity, an extremely strong tendency, to land 3 with long-run relative frequency $1/6$. The small value of $1/6$ does not measure this tendency. According to Giere, the die has a weak tendency to land 3. The value of $1/6$ does measure this tendency. There are other problems in identifying what a propensity actually represents. If we say that a given experiment has a property which causes a particular event to occur with a certain long-run frequency, calling this property a propensity gives us no additional information [Hitchcock 2004]. Single-case propensities have the difficulty that statements about them are untestable, and that they are in the words of Gillies [Gillies 2000], "metaphysical rather than scientific". Another problem is that it is not clear if single-case propensity theories necessarily obey the probability calculus.

Humphreys [Humphreys 1985] argues that propensities do not obey Kolmogorov's probability calculus. Probability calculus implies Bayes' theorem, see equation 2.40, which allows the inversion of conditional probabilities. Propensity measures appear to represent some form of causal relationship, and by their nature causal relationships are asymmetrical and cannot normally be inverted. This leads to what is know as the Humphreys' paradox, that whatever they are, propensities must not obey the usual probability calculus. Popper's axiomatisation of primitive

conditional probabilities does not guarantee that conditional probabilities can be inverted and thus avoids this particular problem.

There seems to be a general lack of clarity on exactly what propensities are, and as such we have a variety of interpretations covering, among other things, relative frequencies, probability values and causal tendencies. So it turns out there is, as yet, no single propensity interpretation of probability.

3.8.5 Subjective Probability

Subjective probability, largely introduced by Frank P. Ramsey [Ramsey 1926] and Bruno de Finetti [De Finetti 1937], sometimes called subjective Bayesianism, equates probability to the belief of a suitable agent. So that in effect the probability of an event is the degree of belief of a suitable agent in the occurrence of that event. So, what constitutes a suitable agent? Allowing any agent is sometimes called unconstrained subjectivism. It has been shown, see [Tversky & Kahneman 1974] among others, that people, when asked to assign probabilities to various events, often break the rules of probability calculus. This means there are no bounds or limits to the probabilities such agents might assign, this limits any possible usefulness of unconstrained subjectivism. Restricting agents to be strictly rational opens more interesting possibilities. Using rational agents Ramsey argued that it was possible to treat probability as a form of logic, the logic of partial belief. A rational agent is required to be logically consistent, this implies that the agent obeys the axioms of probability.

Subjective probabilities are often examined in the context of betting, de Finetti described it as follows:

> Let us suppose that an individual is obliged to evaluate the rate p at which he would be ready to exchange the possession of an arbitrary sum S (positive or negative) dependent on the occurrence of a given event E, for the possession of the sum pS; we will say by definition that this number p is the measure of the degree of probability attributed by the individual considered to the event E, or, more simply, that p is the probability of E (according to the individual considered; this specification can be implicit if there is no ambiguity).
>
> – Bruno de Finetti

Definition 40. In gambling a **Dutch Book** or lock is a set of odds and bets which guarantees a profit, regardless of the outcome of the gamble.

It can be shown, [Skyrms 1984], that if your subjective probabilities violate probability calculus then you are vulnerable to a Dutch book, however, if they are coherent then no Dutch book can be made against you [Kemeny 1955]. If your subjective probabilities are coherent, then they conform to probability calculus.

There are numerous problems with the betting analogy, the agent might not want to give away their estimation of a true probability, they may not wish to bet on a given event, both the cost of the bet and the value of the prize have to be evaluated relative to the wealth of the agent, the divisibility of the coinage being used determines the possible granularity of the subjective probabilities, and there are numerous other possible difficulties. To avoid the issue of granularity the prizes can be measured in utilities which are infinitely divisible, and utility is a linear function of utility.

The relationship between utilities, probabilities and rational preferences has been repeatedly examined. Arnauld and Nicole [Arnauld & Nicole 1662] showed how utilities and probabilities can be used together to determine rational preferences. Ramsey [Ramsey 1926], Savage [Savage 1954] and Jeffrey [Jeffrey 1965] show how to do the reverse, that is to derive both probabilities and utilities from rational preferences. Ramsey showed that, under various assumptions, he could define a real-valued utility function of events. These functions represent the agent's preferences. He established that ratios of utility-differences are invariant to the choice of utility function. Ramsey used this to define degrees of belief as ratios of such differences. Thus if an agent is indifferent between A, and the gamble "B if X, C otherwise." Then it follows from considerations of expected utility that its degree of belief in X, $P(X)$, is given by:

$$P(X) = \frac{u(A) - u(C)}{u(B) - u(C)} \tag{3.19}$$

where $u(A), u(B)$ and $u(C)$ are the utility functions for A, B and C respectively. Ramsey shows that such degrees of belief obey the probability calculus. Ramsey refers to this system as "the logic of partial belief", and it is his goal to incorporate probability into the field of logic. However, it is not clear that he achieved this as their are doubts about how agents fix their preference rankings, his definition of consistency requirements and whether it is always possible to find what he refers to as ethically neutral propositions. Savage developed a similar system using utility

functions which are positive linear transformations of each other, and there is a unique probability function. Thus we could get $U_1 = aU_2 + b$ where U_1 and U_2 are utilities, a and b are constants, and $a > 0$. Jeffrey extended this system and developed a theory of decision according to which rational choice maximises 'expected utility', a certain probability-weighted average of utilities. This can be seen as the core of modern decision theory. Lewis, [Lewis 1986a, Lewis 1995], argues that an agent's degrees of belief are best represented by the probability function belonging to a utility function/probability function pair that best rationalises her behavioural dispositions. There are still many problems associated with this subjectivist view, a good summary of one area of debate is given by Alan Hájek:

> The betting interpretation makes subjective probabilities ascertainable to the extent that an agent's betting dispositions are ascertainable. The derivation of them from preferences makes them ascertainable to the extent that his or her preferences are known. However, it is unclear that an agent's full set of preferences is ascertainable even to himself or herself.
>
> – Alan Hájek

A more complete criticism of preference based probability can be found in [Eriksson & Hájek 2007].

We have shown that probability has a number of possible interpretations. Most of these possibilities are still being actively explored and no single theory currently holds sway as each has something to offer which seems to be necessary as part of the overall picture. Indeed there are many variations on the interpretations which are actively being researched: degrees of incoherence [Schervish et al 2000]; aggregation of the opinions of multiple agents [Seidenfeld et al 1989]; subjective probability ascriptions, sets of possible worlds or sets of centred worlds [Lewis 1979]; issues of appropriateness of conditionalisation for updating rules, such as in the Sleeping Beauty problem [Elga 2000]; and 'scoring rules' for subjective probabilities [Winkler 1996].

3.9 Summary

So, what is causality? The answer still appears to be despite substantial investigation and thought by numerous able people that, in general, it is hard to define. The best most of us can do is to say that

I know it when I see it

– Potter Stewart, Associate Justice of the United States Supreme Court

although we suspect it is both easier to identify and more consistent than the "hard-core pornography" to which he was referring. While we are unable to define causality, we can discuss possible underlying mechanisms which in turn lead to expected properties. We have identified mechanistic causality as the interpretation which most closely matches our views and one which lends itself well to the process of causal discovery. Although causal loops exist, learning them from data requires the data to have certain properties which cannot be guaranteed in any given observational dataset. So, we cannot be sure that causal loops represented in a given dataset, can be learnt. Learning about causal interactions within systems based on their properties which can be observed, relies on a statistical analysis of that observed information. This allows us to make a probabilistic analysis of the most likely model of the underlying system. However, we have seen that there is no single undisputed interpretation of what a probability represents. Fortunately as is the case with causality, we can use the expected properties of probability, and our world view, to enable the search for causal relationships. Most computer users do not understand how a computer works, but they can use its interface to manipulate one to achieve their desired result. Similarly, although we cannot claim to fully understand causality, probability or even what constitutes knowledge, we can use the knowledge we do possess to search for causal relationships. Learning causal relationships is still a difficult task and ultimately, while statistical tests and probabilistic models can help to identify possible relationships, they can only be verified through understanding and experiment on the underlying physical processes.

Chapter 4

Learning Causal Networks

This chapter examines some learners that attempt to learn causal relationships from data. Most learners assume the causal Markov condition and so are attempting to learn a structure that is assumed to be a DAG. There are a number of different methods which can be used to learn a DAG, or more generally a mixed graph, from data. Section 4.1 investigates those learners which attempt to learn the complete network in one go and use Bayes rule as a key part of their learning mechanism. Section 4.2 examines learning the whole network using methods not based on Bayes rule. Section 4.3 examines an alternative approach in which the network is not learnt in one go, but rather a small part at a time. In the context of networks this is usually called constraint-based learning. Lastly section 4.4 is a summary of the issues covered in this chapter and explains which features of the examined learners are used by the LUMIN learner introduced in the following chapter.

4.1 Bayesian Network Learning

BNs are probably the most common current computational representation of causal networks. BNs are visually easy to represent and understand and they can model many important kinds of interaction, and the propagation of effects through a BN is sometimes computationally feasible. BNs allow the testing of "what if" hypotheses by setting the value of some variables and observing the effect this has on the remaining variables. So it is not surprising that a significant amount of effort has gone into learning BNs from data. Learning a BN involves learning both the structure and the NPTs. We focus on learning the structure of the BN which is a DAG.

There is a certain pleasing symmetry in the idea of using Bayesian methods to learn a Bayesian network. The basic idea here would be to use some prior idea of the likelihood of a given network and the available data to determine the network with the maximum *a-posteriori* likelihood. Thus in effect all possible networks are checked and compared to the data to determine which is most likely to produce it. This type of network learner is usually referred to as a search-and-score learner since each possible network is given a score which relates to how well it matches the data, the network with the highest score is the one chosen. The simple description hides a very difficult and recurrent problem with this type of Bayesian analysis, that of computational complexity. With 3 variables there are 25 possible causal networks/models and increasing the number of variables increases the number of possible networks/models to analyse at a rate which is at best $2^{n(n-1)/2}$ where n is the number of variables. It has been shown that in general finding an optimal Bayesian network structure is NP-hard [Chickering et al 1994]. Thus when dealing with this type of problem an attempt is usually made to reduce the number of cases which must be tested.

It is typical when learning a causal BN to require it adheres to the Causal Markov condition, see definition 20 section 2.9.3, and that the BN and the underlying distribution are faithful to each other.

> **Definition 41.** A graph G and distribution P are **faithful** to one another if all and only the independencies entailed by G are present in P.

Its also usually assumed that the variables are discrete, that there are no missing values and no hidden variables. However, methods have been developed to help overcome these limitations. Also there is the potential to limit the possible structures by using knowledge of relationships within the data.

4.1.1 Finding Possible Model Structures

Assume, following the reasoning in [Heckerman et al 1997], we have variables $\mathbf{X} = \{X_1, \cdots, X_n\}$ and data $\mathbf{D} = \{x_1, \cdots, x_n\}$ drawn randomly from some unknown distribution for $\mathbf{X}$, each case x in $\mathbf{D}$ is an observation of all the variables in $\mathbf{X}$. Assume there is a causal model m which represents this domain, and to allow us to model this with a BN we further assume that m has the structure of a directed acyclic graph and that it adheres to the Causal Markov assumption. At this point we do not know the structure or parameters of m. We define $\mathbf{M}$ to be a variable with states m

corresponding to the possible models. We can encode our uncertainty of the actual true model with a probability distribution $p(m)$. Also for each model m we define a continuous vector-valued variable Θ_m whose values θ_m are its possible true parameters. We can encode uncertainty about Θ using the probability density function $p(\theta_m \mid m)$. Using Bayes' rule we get:

$$\begin{aligned} p(m \mid \mathbf{D}) &= \frac{p(\mathbf{D} \mid m)\, p(m)}{\sum_{m'} p(m')\, p(\mathbf{D} \mid m')} \\ p(\theta_m \mid \mathbf{D}, m) &= \frac{p(\theta_m \mid m)\, p(\mathbf{D} \mid \theta_m, m)}{p(\mathbf{D} \mid m)} \end{aligned} \tag{4.1}$$

where

$$p(\mathbf{D} \mid m) = \int p(\mathbf{D} \mid \theta_m, m)\, p(\theta_m \mid m)\, d\theta_m \tag{4.2}$$

is called the marginal likelihood. It is possible to determine the probability that a given hypothesis h is true given $\mathbf{D}$ by averaging over all possible models and parameters

$$\begin{aligned} p(h \mid \mathbf{D}) &= \sum_m p(m \mid \mathbf{D})\, p(h \mid \mathbf{D}, m) \\ p(h \mid \mathbf{D}, m) &= \int p(h \mid \theta_m, m)\, p(\theta_m \mid \mathbf{D}, m)\, d\theta_m \end{aligned} \tag{4.3}$$

At this point we still have the problem of having to deal with all possible models which will in general be infeasible. An assumption that can be made to simplify the computation is that the likelihood term $p(x \mid \theta_m, m)$ can be factored as follows:

$$p(x \mid \theta_m, m) = \prod_{i=1}^{n} p(x_i \mid \mathbf{pa}_i, \theta_i, m) \tag{4.4}$$

where each local likelihood $p(x_i \mid \mathbf{pa}_i, \theta_i, m)$ is in the exponential family, $\mathbf{pa}_i$ denotes the configuration of the variables corresponding to parents of node x_i, and θ_i denotes the set of parameters associated with the local likelihood for variable x_i. Such a factorisation occurs when each variable $X_i \in \mathbf{X}$ is discrete, having r_i possible values, and each local likelihood is a collection of multinomial distributions, one for each configuration of $\mathbf{Pa}_i$:

$$p\left(x_i^k \mid \mathbf{pa}_i^j, \theta_i, m\right) = \theta_{ijk} > 0 \tag{4.5}$$

where $\mathbf{pa}_i^1, \cdots, \mathbf{pa}_i^{q_i}$ $\left(q_i = \prod_{x_i \in \mathbf{Pa}_i} r_i\right)$ denote the configurations of $\mathbf{Pa}_i$ and $\theta_i = \left(\left(\theta_{ijk}\right)_{k=2}^{r_i}\right)_{j=1}^{q_i}$ are the parameters. The parameter θ_{ij1} is given by $1 - \sum_{k=2}^{r_i} \theta_{ijk}$. Now define the vector of parameters

$$\theta_{ij} = (\theta_{ij2}, \cdots, \theta_{ijr_i}) \tag{4.6}$$

for all i and j. Also for efficiency assume all parameters are mutually independent, then given a random sample $\mathbf{D}$ we have

$$p(\theta_m \mid \mathbf{D}, m) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} p(\theta_{ij}, \mid \mathbf{D}, m) \tag{4.7}$$

So each vector of parameter θ_{ij} can be updated independently. Assuming each vector θ_{ij} has a conjugate prior, see [Bernardo and Smith 1994], that is a Dirichlet distribution of the form $Dir(\theta_{ij} \mid \alpha_{ij1}, \cdots, \alpha_{ijr_i})$ gives a posterior distribution for the parameters

$$p(\theta_{ij} \mid \mathbf{D}, m) = Dir(\theta_{ij} \mid \alpha_{ij1} + N_{ij1}, \cdots, \alpha_{ijr_i} + N_{ijr_i}) \tag{4.8}$$

where N_{ijk} is the number of cases in $\mathbf{D}$ in which $X_i = x_i^k$ and $\mathbf{Pa}_i = \mathbf{pa}_i^j$. The collection of counts N_{ijk} are sufficient statistics of the data for the model m. The marginal likelihood would then be

$$p(\mathbf{D} \mid m) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij} + N_{ij})} \cdot \prod_{k=1}^{r_i} \frac{\Gamma\left(\alpha_{ijk} + N_{ijk}\right)}{\Gamma\left(\alpha_{ijk}\right)} \tag{4.9}$$

where $\alpha_{ij} = \sum_{k=1}^{r_i} \alpha_{ijk}$ and $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$ and $\Gamma(n) = (n-1)!$ and there are q_i unique instantiations of $\mathbf{pa}_i$. It is then possible to calculate the posterior probabilities $p(m \mid \mathbf{D})$ using equations 4.1 and 4.9.

4.1.2 Finding Possible Model Structures - A More Heuristic Approach

The approach in section 4.1.1 assumes little knowledge of any relationships in the data. The Bayesian way of including prior knowledge would be in the priors, this is shown in section 4.1.3. An alternative which is commonly used even with otherwise Bayesian methods is that of node ordering. Node ordering does not provide information about the existence of relationships between variables, but is an indication to cause an effect between two related variables. We will outline the **K2** algorithm, [Cooper & Herskovits 1992], which uses a more heuristic approach to building a network from data. The **K2** algorithm looks for the model structure, m_s, with the

maximum posterior probability by calculating the ratio

$$\frac{p\left(m_{s_i} \mid \mathbf{D}\right)}{p\left(m_{s_j} \mid \mathbf{D}\right)} = \frac{\frac{p\left(m_{s_i}, \mathbf{D}\right)}{p(\mathbf{D})}}{\frac{p\left(m_{s_j}, \mathbf{D}\right)}{p(\mathbf{D})}} = \frac{p\left(m_{s_i}, \mathbf{D}\right)}{p\left(m_{s_j}, \mathbf{D}\right)} \tag{4.10}$$

As before we assume the variables $\mathbf{X}$ are discrete, each data point is independent and there are no missing values, then we have

$$p\left(m_s, \mathbf{D}\right) = \int_{m_s} p\left(\mathbf{D} \mid m_s, \theta_m\right) f\left(\theta_m \mid m_s\right) p\left(m_s\right) d\theta_m \tag{4.11}$$

where θ_m is defined as before, and f is the conditional probability density function over θ_m given m_s. Cooper and Herskovits refer to $p\left(m_s\right)$ in equation 4.11 as a form of preference bias [Utgoff 1986]. It follows then that

$$p\left(m_s, \mathbf{D}\right) = \int_{\theta_m} \left[\prod_{h=1}^{n} p\left(C_h \mid m_s, \theta_m\right)\right] f\left(\theta_m \mid m_s\right) p\left(m_s\right) d\theta_m \tag{4.12}$$

where n is the number of cases in $\mathbf{D}$ and C_h is the hth case in $\mathbf{D}$. From the same assumptions in section 4.1.1 we have

$$N_{ij} = \sum_{k=1}^{r_i} N_{ijk} \tag{4.13}$$

which leads to

$$p\left(m_s, \mathbf{D}\right) = p\left(m_s\right) \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}! \tag{4.14}$$

compare this to equation 4.9, not surprisingly the same assumptions lead to similar conclusions. At this point we still have the usual problem of having to search through too many possible networks and we need to take steps to reduce the number of possible models. We introduce a node ordering such that if x_i precedes x_j in the ordering then it is not allowed to have an arc from x_j to x_i. While restrictive, in terms of the allowed relationships between any two variables, this is much less so in terms of the overall structure than other methods of simplification such as the imposition of a tree structure [Chow & Liu 1968], see section 4.2. Then by adding in uniform priors for the networks we get

$$p\left(m_s, \mathbf{D}\right) = c \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}! \tag{4.15}$$

where c is the constant prior probability of each $p(m_s)$. Equation 4.15 can be maximised using the second inner product, that is

$$\max_{m_s}[p(m_s,\mathbf{D})] = c\prod_{i=i}^{n}\max_{\mathbf{pa}_i}\left[\prod_{j=1}^{q_i}\frac{(r_i-1)!}{(N_{ij}+r_i-1)!}\prod_{k=1}^{r_i}N_{ijk}!\right] \tag{4.16}$$

Assuming a node has no parents and then adding those parents which most increase the probability of the resulting structure and stopping when no further increase can be made. Using the following equation in a greedy search

$$g(i,\mathbf{pa}_i) = \prod_{j=1}^{q_i}\frac{(r_i-1)!}{(N_{ij}+r_i-1)!}\prod_{k=1}^{r_i}N_{ijk}! \tag{4.17}$$

and defining the function $Pred(x_i)$ that returns the set of nodes that precedes x_i, Cooper and Herskovits define the **K2** algorithm which has proved to be a very efficient algorithm.

Algorithm 4.1 The K2 Algorithm

1. **procedure** K2;
2. {Input: A set of n nodes, an ordering on the nodes, an upper bound u on the number of parents a node may have, and a database D containing m cases. }
3. {Output: For each node, a printout of the parents of the node.}
4. **for** i := 1 **to** n **do**
   (a) $\mathbf{pa}_i$: = $\emptyset$;
   (b) P_{old} := $g(i,\mathbf{pa}_i)$; {This function is computed using equation (4.17).}
   (c) OKToProceed := true
   (d) **while** OKToProceed **and** $|\mathbf{pa_i}| < u$ **do**
      i. let z be the node in $Pred(x_i) - \mathbf{pa}_i$ that maximises $g(i,\mathbf{pa}_i \cup \{z\})$;
      ii. P_{new} := $g(i,\mathbf{pa}_i \cup \{z\})$;
      iii. **if** $P_{new} > P_{old}$ **then**
         A. P_{old} := P_{new};
         B. $\mathbf{pa}_i$:= $\mathbf{pa}_i \cup \{z\}$
      iv. **else** OKToProceed := false;
   (e) **end** {while};
   (f) write('Node:', x_i, 'Parents of this node: $\mathbf{pa}_i$)
5. **end** {for};
6. **end** {K2};

The above discussion assumed that the dataset was complete, that is, there were no missing values, and that it contained all the variables required to generate the data. It is sometimes the case that either or both of these conditions are not met. Let C_h denote the set of variables in the hth case that have known values and let C'_h denote the set of variables in the hth case with missing values. We can then compute the probability of the hth case as

$$p(C_h \mid m_s, \theta_m) = \sum_{C'_h} p\left(C_h, C'_h \mid m_s, \theta_m\right) \tag{4.18}$$

where $\sum_{C'_h}$ indicates running through all the possible values of the variables in C'_h. This assumes that the database is sufficiently complete or otherwise there is enough information to know all the possible values in the the set C'_h. Using the result of equation 4.18 in equation 4.12 we get

$$p(m_s, \mathbf{D}) = \int_{\theta_m} \left[\prod_{h=1}^{n} \left[\sum_{C'_h} p\left(C_h, C'_h \mid m_s, \theta_m\right) \right] \right] f(\theta_m \mid m_s)\, p(m_s)\, d\theta_m \tag{4.19}$$

Let x_i be a variable in $\mathbf{D}$ then we can define assignment of a value to the variable as $x_i = d_{ih}$ where d_{ih} is the value of the variable in the hth case. So if x_i is in C'_h then the value d_{ih} is not known. In equation 4.18 for each x_i in C'_h we have d_{ih} assume every possible value in turn. Using this notation we can rewrite equation 4.19 as

$$p(m_s, \mathbf{D}) = \sum_{C'_1} \cdots \sum_{C'_l} \int_{\theta_m} \left[\prod_{h=1}^{l} p\left(C_h, C'_h \mid m_s, \theta_m\right) \right] (\theta_m \mid m_s)\, p(m_s)\, d\theta_m \tag{4.20}$$

This is just a sum of the type of integrals in equation 4.12 which can therefore be solved by repeated application of equation 4.14. Like many instances of Bayesian analysis while this is a perfect theoretical solution it is computationally infeasible for many cases as its computational effort is exponential in the number of missing values.

In addition to assuming the dataset was complete we previously also assumed that all the variables responsible for generating the data were part of the dataset. When this is not the case we are said to have hidden, or latent, variables. While it may appear that nothing is known about hidden variables, we can view them as normal variables whose values are never known. In this situation, providing we have knowledge of the possible values for these variables, we already have a sound solution in equation 4.20. In those circumstances when the possible values of the hidden variables are not known, it may be possible to learn them, or at least their number, using

unsupervised learning techniques [Cheesman et al 1988]. An issue with hidden variables is that for any dataset there are an infinite number of possible generating networks containing any number of hidden variables. In order to manage this problem some method of limiting the possible number of hidden variables has to be used. One option is to use a simple fixed number of allowed hidden variables, alternatively a search with a parametrised number of hidden variables can be used and it may also be possible to use statistical tests to determine when hidden variables are likely to be present [Pearl and Verma 1991, Spirtes & Glymour 1990, Verma and Pearl 1990].

4.1.3 Finding Priors for the Model Parameters

The next issue to tackle is that of the priors for the both the structure and parameters, that is, $p(m)$ and $p(\theta_m \mid m)$. Similar issues confront us here as with the structure and parameters, in general there will be too many possibilities to perform an exhaustive search and so we need to make some simplifying assumptions. [Heckerman 1995a] when considering the parameter priors assumes that the parameters are independent and local likelihoods are multinomial distributions. Two additional concepts are required, that of Markov equivalence and distribution equivalence.

> **Definition 42.** If two model structures for **X** entail the same conditional-independence assertions then they are said to be **Markov equivalent** [Verma and Pearl 1990].

An alternative description is that two model structures are Markov equivalent if and only if they have the same structure ignoring arc directions and the same v-structure.

> **Definition 43.** A **v-structure** is an ordered tuple (X,Y,Z) such that there is an arc from X to Y and from Z to Y , but no arc between X and Z.

Complete model structures on **X** are also Markov equivalent.

> **Definition 44.** A **complete model structure** is one in which there are no missing edges and which encodes no assertion of conditional independence.

Distribution equivalence is related to Markov equivalence.

> **Definition 45.** Suppose all the causal models for **X** under consideration have local likelihoods in the family $\mathcal{F}$. We say that two model structures m_1 and m_2 for **X** are **distribution equivalent** with respect to $\mathcal{F}$ if they represent the same joint probability distributions for **X**. Thus for every θ_{m1} there exists a θ_{m2} such that $p(x \mid \theta_{m1}, m_1) = p(x \mid \theta_{m2}, m_2)$.

Distribution equivalence with respect to some $\mathcal{F}$ implies Markov equivalence, but the converse does not hold in general. Assuming parameter independence and distribution equivalence, implies that the parameters for any complete model structure m_c must have a Dirichlet distribution, and [Geiger & Heckerman 1995] show that the hyperparameters have constraints given by

$$\alpha_{ijk} = \alpha p\left(x_i^k, \mathbf{pa}_i^j \mid m_c\right) \tag{4.21}$$

where α is the user's equivalent of sample size, and $p\left(x_i^k, \mathbf{pa}_i^j \mid m_c\right)$ is computed from the user's joint probability distribution $p(x \mid m_c)$. This idea can be extended to cover incomplete model structures with the additional assumption of parameter modularity. Parameter modularity states that if X_i has the same parents in model m_1 and m_2 then

$$p(\theta_{ij} \mid m_1) = p(\theta_{ij} \mid m_2) \tag{4.22}$$

for $j = 1, \cdots, q_i$. This determines that the distributions for parameters θ_{ij} depend only on the structure of the model that is local to variable X_i, that is, X_i and its parents. So, with the assumptions of parameter independence, parameter modularity and non-zero priors it is possible to construct priors for the parameters of an arbitrary model structure given the priors on complete model structures. Parameter independence allows for the separate construction of the priors for each node. If a node X_i has parents $\mathbf{Pa}_i$ in a given model structure we find a complete structure in which the node has the same parents. Then using equation 4.21 and parameter modularity we can construct the parameter priors for this node. So, from α and $p(x \mid m_c)$ we can derive the parameter priors for all possible structures. The distribution $p(x \mid m_c)$ can be assessed by constructing a causal model, called an *a-priori* model, that encodes it. [Heckerman et al 1995] discuss the construction of such a model.

The issue of priors does not relate only to the parameters, but also to the model structures. There are a number of simple common practises used when facing this problem, the simplest solution is to assign the same prior to all models[1]. However, it may be the case that sufficient knowledge is available to eliminate some structures and more accurately reflect the priors of others. In this case it is possible to allow the user to define the priors. There may be too many possible structures for each to be assigned a prior and some compromise may be required. It

[1]This is really use of the principle of indifference, see section 3.8.1 for its definition and mention of possible problems with its use.

is possible to automate the assignment of priors by making sufficient assumptions about the structure [Buntine 1991]. The assumptions are that the variables can be ordered, restricting the possible direction of an arc, and that the presence or absence of possible arcs are mutually independent. Given these assumptions $n(n-1)/2$ probability assessments determines the prior probability of every possible model structure. There are numerous alternative schemes for generating priors that have been proposed including using imaginary data [Good 1950], from a domain expert [Madigan et al 1995], using a prior model [Heckerman 1995a], using domain knowledge to specify a prior distribution for parameters of a logistic regression model [Dayanik et al 2006], and using Laplace approximations [Gelfand & Dey 1994].

4.1.4 Model Selection and Selective Model Averaging

The above methods aim to reduce the number of structures and priors which need to be examined to find the optimal match to the data. It is possible that even if a perfect model exists it is not among those remaining as the simplifying assumptions may be incorrect. It is also possible that no perfect model exists given the data, this can be due to missing data items, hidden variables or the dataset being too small or not representative of the underlying distribution. Even if a good, or perfect, model exists within the remaining model selections, after the simplifying assumptions, there will usually be more cases than can reasonably be tested using equation 4.1. There are two common approaches used to further reduce the task of choosing the best model to fit the data, these are *model selection* and *selective model averaging*.

Model Selection is simply the process of choosing one good model and treating it as though it was the perfect model. In some ways this is analogous to part of the process of the EM algorithm, see section 2.9.2. Choosing a good model requires some from of scoring function to compare the model to the dataset, and some method of choosing which models to test and what threshold to accept as good. There are a number of Bayesian scoring functions the first being K2 [Cooper & Herskovits 1992] which was generalised into the Bayesian Dirichlet or BD score [Heckerman et al 1995].

Definition 46. The **Bayesian Dirichlet Score** is defined as:

$$BD(m,\mathbf{D}) := \log\left(p_{pr}(m)\right) + \sum_{i=1}^{n}\left[\sum_{j=1}^{q_i}\left[\log\left(\frac{\Gamma(\alpha_{ij})}{\Gamma(N_{ij}+\alpha_{ij})}\right) + \sum_{k=1}^{r_i}\log\left(\frac{\Gamma\left(N_{ijk}+\alpha_{ijk}\right)}{\Gamma\left(\alpha_{ijk}\right)}\right)\right]\right] \tag{4.23}$$

where $p_{pr}(m)$ is the prior probability of the structure of m, this is closely related to equation 4.9.

Adding the assumption of distribution equivalence leads to a alternate expression of the hyper-parameter η, which is easier to calculate

$$\eta_{ijk} = \eta \times p\left(x_i^k, \mathbf{pa}_i^j \mid m_0\right) \tag{4.24}$$

where $p(. \mid m_0)$ represents a probability distribution associated with a prior Bayesian network m_0 and η is a parameter representing equivalent sample size. Assuming an equal probability for each configuration of a node's parents so that $p(x_{ik}, w_{ij}) = \frac{1}{r_i q_i}$ leads to a scoring function, *BDeu*, first proposed by [Buntine 1991].

Definition 47. The Bayesian Dirichlet, likelihood equivalence, uniform joint distribution score, **BDeu Score** is defined as:

$$BDeu(m,\mathbf{D}) := \log\left(p_{pr}(m)\right) + \sum_{i=1}^{n}\left[\sum_{j=1}^{q_i}\left[\log\left(\frac{\Gamma\left(\frac{\eta}{q_i}\right)}{\Gamma\left(N_{ij}+\frac{\eta}{q_i}\right)}\right) + \sum_{k=1}^{r_i}\log\left(\frac{\Gamma\left(N_{ijk}+\frac{\eta}{r_i q_i}\right)}{\Gamma\left(\frac{\eta}{r_i q_i}\right)}\right)\right]\right] \tag{4.25}$$

Another common Bayesian scoring function is the Bayesian Information Criterion, BIC, derived in [Schwarz 1978]. This measure uses the Gaussian approximation that for large data sets

$$p(\theta_m \mid \mathbf{D}, m) \propto p(\mathbf{D} \mid \theta_m, m) \cdot p(\theta_m \mid m) \tag{4.26}$$

Definition 48. We then define

$$g(\theta_m) := \log\left(p(\mathbf{D} \mid \theta_m, m) \cdot p(\theta_m \mid m)\right) \tag{4.27}$$

The **Bayesian Information Criterion** can be represented as

$$\log p(\mathbf{D} \mid m) \approx \log p\left(\mathbf{D} \mid \hat{\theta}_m, m\right) - \frac{d}{2}\log N \tag{4.28}$$

where $\hat{\theta}_m$ is the maximum likelihood configuration of θ_m, that is, the configuration of θ_m that maximises the value of $g(\theta_m)$, and d is the number of free parameters to be estimated, that is, the dimension of $g(\theta_m)$.

The BIC measure is interesting for a number of reasons: its intuitive in that it looks for a model which best represents the data, $p\left(\mathbf{D} \mid \hat{\theta}_m, m\right)$, and penalises complex models, $-\frac{d}{2}\log N$; it does not depend on the prior, hence there is no need to calculate it; it is exactly equal to the minimum description length criterion, [Rissanen 1987], but with negative sign; it is related to other penalised likelihood criteria such as the Akaike information criterion.

It is common for the search through the model space to use a local search that makes one change to the current model by adding, deleting or inverting the direction of an arc then testing the score of the new model. The search terminates either when some threshold is met or after a given time or number of iterations. This technique has been shown to produce reasonably accurate predictions [Cooper & Herskovits 1992, Aliferis and Cooper 1994, Heckerman et al 1995]. A common problem with building a single model is that of overfitting. The model will accurately reflect the training data, but may not be such a good representation of the data distribution as a whole. In other fields of ML the technique of using a number of closely related models and combining the result is used, see section 2.17. Model averaging is the BN version of an ensemble technique. A number of likely models, $m_1, \ldots, m_t$ are selected. A test of the posterior probability of each of the models given the data is used to generate a set of normalised weights $w_1, \ldots, w_t$, where $w_i = C_n p\left(m_i \mid \mathbf{D}\right)$ and C_n is a constant chosen so that $w_1 + \cdots + w_t = 1$. The aim is to find the true distribution Y of some target variable. The models each produce a prediction $\hat{Y}$. The true distribution is approximated by the weighted average $Y = \sum_{i=1}^{t} w_i \hat{Y}_i$ of each of the models.

4.2 Non-Bayesian Network Learners

Just as there are Bayesian-based algorithms for learning which network was most likely to produce a given dataset, there are also non-Bayesian search-and-score, whole network, learning algorithms. The major issues here are the same as in the Bayesian search case, that is the vast space of possible networks to search and how to quantify how close a given network is to the desired one.

Search-and-score is a classic ML approach to learning [Wilson 1970, Quinlan 1986], the basic idea is that there is some network responsible for producing the dataset and all that is required is to find it from the set of possible networks. So, what is required is some measure of how close a given network is to the required one and a method of choosing the next network to test. Just as with the Bayesian approach an issue is the size of the search space for all non-trivial

datasets. Since an exhaustive search is not feasible some method of limiting the search space is required.

[Chow & Liu 1968] searched for the best *tree structured* network, that is, where each node has at most one parent. They use information measures to determine the best fit, and are able to reduce the search space as follows. Assume we are given a distribution $p(x_1, x_2, \ldots, x_n)$ of discrete variables x_i. We want to find a tree τ with probability distribution $p_\tau(x_1, x_2, \ldots, x_n)$ such that

$$I_c(p, p_\tau) \leq I_c(p, p_t) \forall t \in T \tag{4.29}$$

where T is the set is of all possible first-order dependence trees, and $I_c(p_a, p_b)$ is the closeness of approximation [Lewis 1959] between probability distributions p_a and p_b, over the set of discrete variables $\mathbf{x} = (x_1, x_2, \ldots, x_n)$.

Definition 49. The **closeness of approximation** $I_c(p_a, p_b)$is defined as

$$I_c(p_a, p_b) := \sum_{\mathbf{x}} p_a(x) \log \frac{p_a(x)}{p_b(x)} \tag{4.30}$$

We can see this is identical to the Kullback-Leibler divergence, see definition 7. Thus it is clear that mutual information and closeness of approximation are closely related. Each branch of a dependence tree can be assigned a weight $I_c\left(x_i, x_{j(i)}\right)$. A maximum-weight dependence tree is a dependence tree t such that for all t' in T

$$\sum_{i=1}^{n} I_c\left(x_i, x_{j(i)}\right) \geq \sum_{i=1}^{n} I_c\left(x_i, x_{j'(i)}\right) \tag{4.31}$$

A probability distribution of tree dependence $p_t(x)$ is an optimum approximation to $p(x)$ if and only if its dependence tree t has maximum weight. From equation 4.30 we have

$$\begin{aligned} I_c(p, p_t) &= -\sum_x p(x) \sum_{i=1}^{n} \log p\left(x_i \mid x_{j(i)}\right) + \sum_x p(x) \log p(x) \\ &= -\sum_x p(x) \sum_{i=1, j(i)\neq 0}^{n} \log \frac{p\left(x_i, x_{j(i)}\right)}{p(x_i)\, p\left(x_{j(i)}\right)} - \sum_x p(x) \sum_{i=1}^{n} \log p(x_i) + \sum_x p(x) \log p(x) \end{aligned} \tag{4.32}$$

Given that $p(x_i)$ and $p\left(x_i, x_{j(i)}\right)$ are components of $p(\mathbf{x})$ we have

$$-\sum_x p(x)\log p(x_i) = \sum_{x_i} p(x_i)\log p(x_i) \tag{4.33}$$

this is simply the entropy of x_i usually written as $H(x_i)$, also

$$\begin{aligned}\sum_x p(x)\log\frac{p\left(x_i, x_{j(i)}\right)}{p(x_i)\,p\left(x_{j(i)}\right)} &= \sum_{x_i, x_{j(i)}} p\left(x_i, x_{j(i)}\right)\log\frac{p\left(x_i, x_{j(i)}\right)}{p(x_i)\,p\left(x_{j(i)}\right)} \\ &= I\left(x_i; x_{j(i)}\right)\end{aligned} \tag{4.34}$$

So, we can rewrite equation 4.32 as

$$I_c(p, p_t) = -\sum_{i=1}^{n} I\left(x_i; x_{j(i)}\right) + \sum_{i=1}^{n} H(x_i) - H(\mathbf{x}) \tag{4.35}$$

Now $H(\mathbf{x})$ and $H(x_i)$ are independent of the dependence tree and $I_c(p, p_t)$ is non negative, minimising equation 4.35 is the same as maximising $\sum_{i=1}^{n} I\left(x_i; x_{j(i)}\right)$. While this only applies to a limited type of BN it has been proved that the Chow-Liu algorithm will find the tree structured network with a distribution closest to the one supplied, and that if the supplied distribution can be produced by a tree structured BN it will find the correct BN. The algorithm is also very efficient working in $O\left(N^2\right)$ time as only pair wise dependency calculations are required. The Chow-Liu algorithm was possibly the first of this type of network learning algorithm. There have since been other more comprehensive algorithms developed which use an information based measure to score networks including but not limited to [Rebane & Pearl 1987, Herskovits & Cooper 1990, Wong & Xiang 1994, Lam & Bacchus 1994, Suzuki 1996, de Campos 2006]. Like their Bayesian counterparts these algorithms vary in their assumptions, requirements and in the type of network they produce. [Rebane & Pearl 1987] considered only polytrees[2], [Herskovits & Cooper 1990] considered Bayesian Networks in general, but required the additional information of node ordering to produce directed edges and [Lam & Bacchus 1994] produce Bayesian Networks with directed edges without needing node ordering. We will be examining this algorithm in more detail.

Lam and Bacchus have an unusual starting point, not in their methodology which can be seen as an extension of earlier work, [Chow & Liu 1968, Rebane & Pearl 1987], but in that their goal is a simple network which generates a probability distribution close to that supplied, rather than

[2] A polytree is a directed graph with at most one undirected path between any two vertices.

the exact reproduction of the generating network. Their methodology is simply that of search-and-score, however, the score is based on the Minimum Description Length, MDL, principle [Rissanen 1978]. In this context they are looking for the model which minimises the length of encoding of the model itself and the length of encoding of the data given the model. A BN uses a list of the parents of each node and a set of conditional probabilities associated with each node. Assuming there are n nodes, a node with k parents would need $k\log_2(n)$ bits to encode its parents. In addition to encode the NPT we need the product of the number of bits required to encode each conditional probability and the number of conditional probabilities required. Since the sum of the conditional probabilities must add to 1 we need to encode all bar one of them, the last being inferred. So, if a node x_i has k_i parents represented by the set $\mathbf{pa}_i$, and can take on s_i values, then we can define the total description length of a network to be

$$\sum_{i=1}^{n}\left[k_i\log_2(n)+d(s_i-1)\prod_{j\in\mathbf{pa}_i}s_j\right] \tag{4.36}$$

where d represents the number of bits needed to store a value. Since adding connections increases both the number of parents and the size of the NPT this scheme will favour sparsely connected networks. Now to encode the data it is possible to assign a unique binary string to each distinct database state, or event, that is to each distinct instantiation of all the nodes in the network. To minimise the length of the database description we need the most probable states to have the shortest identifying strings. This problem has already been solved, with Huffman's algorithm generating Huffman codes [Cormen et al 1990]. If each event e_i occurs with a probability of p_i then Huffman's algorithm will assign it a string of length approximately $-\log_2(p_i)$ [Lelewer & Hirschberg 1987]. With N data points, assuming N is large, we would expect there to be Np_i occurrences of each event e_i. So the length of the string encoding the database would be approximately

$$-N\sum_i p_i\log_2(p_i) \tag{4.37}$$

One problem here is that the values of p_i are the true probabilities of the underlying distribution and these values will not in general be known. So, instead of the true probability of an event p_i we could use the probability of the event as generated by a learnt BN, q_i. So the length of the

string encoding the database would then be approximately

$$-N\sum_i p_i \log_2(q_i) \tag{4.38}$$

Since this equation still requires p_i this will have to be estimated from the frequency of occurrence of each event in the database. Lam and Bacchus use the Kullback-Leibler cross-entropy to determine how close the probability distribution defined by a given network is to that in the database. The Kullback-Leibler divergence, sometimes called the Kullback-Leibler cross-entropy, is defined by equation 2.8 in section 2.3. If P and Q are probability distributions defined over the same event space then we represent the Kullback-Leibler divergence as $D_{KL}(P \parallel Q)$. Chow and Liu [Chow & Liu 1968] developed a theorem that if the mutual information, see equation 2.6, between any two nodes X_i and X_j was used as a weight on the arc between the nodes then the cross-entropy $D_{KL}(P \parallel Q)$ over all tree structured distributions Q is minimised when the structure representing $Q(\mathbf{X})$ is a maximum weight spanning tree. Lam and Bacchus extend this by defining a new measure for a node X_i, and an arbitrary set of parents

$$W(X_i, \mathbf{pa}_i) = \sum_{X_i, \mathbf{pa}_i} P(X_i, \mathbf{pa}_i) \log_2 \left(\frac{P(X_i, \mathbf{pa}_i)}{P(X_i) P(\mathbf{pa}_i)} \right) \tag{4.39}$$

where the sum is over all the possible values that X_i and its parents can take. They then prove that $D_{KL}(P \parallel Q)$ is a monotonically decreasing function of the sum of these weights over all nodes and parent sets, that is of

$$\sum_{i=1, \mathbf{pa}_i \neq \emptyset}^{n} W(X_i, \mathbf{pa}_i) \tag{4.40}$$

Thus the cross entropy is minimised when the network weight is maximised, as they point out this in itself is not very useful as a complete graph[3] will always have maximum weight. However, since they use MDL as part of the measure of how good a learnt network is the extra encoding required for more complex networks will tend to favour simple solutions. Algorithm 4.2 works by dividing the search into sets of networks with an equal number of arcs. If there are n nodes in the network this means that there are networks with between 0 and $n(n-1)/2$ arcs. This is done as it is still computationally infeasible to examine all possible networks, the algorithm thus allows for equal time or effort to be spent evaluating networks across the range of complexity. The PAIRS list is generated before the algorithm is executed and consists of a sorted list of

[3] See definition 44 section 4.1.3.

Algorithm 4.2 Lam and Bacchus Network Learning

1. Remove the element with greatest heuristic value from the S_i's OPEN list and copy it onto the CLOSED list. Let the element's network be G_{old} and the element's pair of nodes be (X_i, X_j).
2. Invoke the PD-procedure on G_{old} and (X_i, X_j) to get a new network G_{new}. The PD-procedure, described in detail below, adds an arc between the nodes X_i and X_j creating a new network G_{new}. It decides on the direction of this new arc, i.e., if it should be $X_i \rightarrow X_j$ or $X_j \rightarrow X_i$, picking the direction that most increases the network's accuracy. In the process it might also reverse the direction of other arcs in G_{old}. Note that G_{new} is a network with $i+1$ arcs, so it must be placed inside the search set S_{i+1} not back into S_i.
3. If G_{new} is fully connected, we place a copy of it into a list of final candidate networks, FINAL.
4. Next we make a new search element consisting of G_{new} and the first pair of nodes from PAIRS that appear after the old pair (X_i, X_j) and between which an arc could be added without generating a cycle in G_{new}. Then, we insert this element into the OPEN list of the search set S_{i+1}, placed in correct order according to the heuristic function.
5. Finally, we make a new search element consisting of G_{old} and the first pair of nodes from PAIRS that appear after the old pair (X_i, X_j) and between which an arc could be added without generating a cycle in G_{old}. This element is inserted into the OPEN list of search set S_i, placed in the correct order according to the heuristic function.

distinct node pairs in descending order of their mutual information. The heuristic is the MDL of the learnt network, that is the sum of equations 4.36 and 4.38 where lower values are preferred. The PD or parents-detection procedure referred to is shown in algorithm 4.3. Overall this is quite an impressive algorithm as it has been shown to perform reasonably well, it can recover both the structure and the direction of the arcs without the need for any additional information.

The non-Bayesian search-and-score learners often use one of a closely related set of information theoretical measures, namely entropy, mutual information, and the Kullback-Leibler divergence [Kullback & Leibler 1951]. It is interesting to note that the usual scoring metrics used by search-and-score learners, AIC, BIC, MDL etc, do not measure the actual accuracy of the learnt network, but they do provide a relative measure of the accuracy with respect to the data. Metrics have been developed which give an absolute accuracy estimate for a given network with respect to a dataset, [Pappas & Gillies 2002], but we are not aware of their use in learners. These measures, sometimes combined with conditional independence tests, have proved able to recover all or part of the structure of a network given only the data the network produces, although determining the direction of causal influence sometimes requires additional information like node

Algorithm 4.3 PD procedure

Input : A network G_{old}. A pair of nodes (X_i, X_j) between which an arc is to be added.

Output : A new network G_{new} with the arc added and some other arcs possibly reversed.

1. Create a new network by adding the arc $(X_i \rightarrow X_j)$ to G_{old}. In this new network we then search locally to determine if we can increase its weight by reversing the direction of some of its arcs. This is accomplished via the following steps.
   (a) Determine the optimal directionality of the arcs attached directly to X_j by examining which directions maximise the weight measure. Some of these arcs may be reversed by this process.
   (b) If the direction of an existing arc is reversed then perform the above directionality determination step on the other node affected.
2. Repeat the above steps except this time with the new network formed by adding the arc $(X_j \rightarrow X_i)$ to G_{old}.
3. Select the network of greatest weight from the two networks found in the above steps. This network is the output.

ordering. Unlike the Bayesian-based methods they can be computationally tractable for large datasets, but are not always guaranteed to produce the correct network[4].

4.3 Constraint-Based Learners

Constraint-based learners attempt to learn a network from data, but unlike the search-and-score model rather than attempt to learn the entire network in one go, they usually try to learn the relationships between subsets of the variables involved. This approach allows them to build a network in a number of small stages. The advantage of this approach is that it is potentially much simpler to learn a limited part of the network and build up the complete network than it is to learn the whole network in one go, but doing so might result in a network that is not as good a representation of the data.

Since most causal network learners assume the goal is a BN there are two tasks to perform: to learn the structure of the network; and to learn the NPTs, see section 2.9.3. In the context of constraint-based learners these two tasks are separate. Under the conditions that are normally assumed to be true for constraint-based learners that task of learning the NPTs given the correct structure is fairly simple [Cooper & Herskovits 1992], so we cover only the

[4]Even assuming the probability distribution can be represented by a DAG.

learning of the structure. There are numerous constraint-based learning algorithms including, TETRAD [Glymour et al 1987] extended in [Scheines et al 1994], IC [Pearl and Verma 1991], PC [Spirtes et al 1991], FCI [Spirtes et al 1993], LCD [Cooper 1997] which was extended in [Silverstein et al 2000], SLA/SLA-Π and TPDA/TPDA-Π [Cheng et al 2002]. They share a number of assumptions and in general search for (in)dependencies and conditional (in)dependencies in the data to determine both the structure and direction of influence within the underlying system that generated the data. We will cover some of these algorithms in detail.

4.3.1 The IC Algorithm

The IC-Algorithm, or inductive causation algorithm, is shown in algorithm 4.4. We give a quick

Algorithm 4.4 IC-Algorithm

Input: $\hat{P}$ a sampled distribution
Output: $core\left(\hat{P}\right)$ a marked hybrid acyclic graph

1. For each pair of variables a and b, search for a set S_{ab} such that $(a;S_{ab};b)$ is in $I\left(\hat{P}\right)$, namely a and b are independent in $\hat{P}$, conditioned on S_{ab}. If there is no such S_{ab}, place an undirected link between a and b.
2. For each pair of non-adjacent variables a and b with a common neighbour c, check if $c \in S_{ab}$. If it is, then continue. If it is not, then add arrowheads pointing at c, (i.e. $a \rightarrow c \leftarrow b$).
3. Form $core\left(\hat{P}\right)$ by recursively adding arrowheads according to the following two rules: If $\overline{ab}$ and there is a strictly directed path from a to b then add an arrowhead at b. If a and b are not adjacent but $\overrightarrow{ac}$ and $c - b$, then direct the link $c \rightarrow b$.
4. If a and b are not adjacent but $\overrightarrow{ac}$ and $c \rightarrow b$ or $c - b$, then mark the link $c \mapsto b$.

Notation: $\overline{ab}$ denotes one of $a - b$, $a \rightarrow b$, $a \leftarrow b$ or $a \leftrightarrow b$ and $\overrightarrow{ab}$ denotes either $a \rightarrow b$ or $a \leftrightarrow b$. $a \mapsto b$ denotes definite causal influence from a to b.

run through of the theory behind this algorithm as the theory and background apply to many constraint-based algorithms. IC assumes the causal model that generates the data is a DAG, D over a set of variables $\mathbf{U}$.

> **Definition 50.** A **causal theory**, T, is defined as $T = \langle D, \Theta_D \rangle$ where Θ_D is a set of parameters compatible with D. Θ_D defines a function for each node $x_i = f_i\left[\mathbf{pa}\left(x_i\right), \varepsilon_i\right]$ and probability measure g_i to each $x_i \in \mathbf{U}$, where $\mathbf{pa}\left(x_i\right)$ are the parents of x_i in D and ε_i is a random disturbance distributed according to g_i. $P(T)$ is the joint probability distribution defined by T.

Definition 51. A **latent structure**, L, is defined as a $L = \langle D, \mathbf{O} \rangle$ where $\mathbf{O}$ is a set of observable variables such that $\mathbf{O} \subseteq \mathbf{U}$.

Definition 52. One latent structure $L = \langle D, \mathbf{O} \rangle$ is **preferred** to another $L' = \langle D', \mathbf{O} \rangle$ written as $L \preceq L'$ *iff* D' can mimic D over $\mathbf{O}$, that is:

$$\forall \Theta_D \, \exists \Theta'_{D'} : \; P_{\mathbf{O}} \left(\left\langle D', \Theta'_{D'} \right\rangle \right) = P_{\mathbf{O}} \left(\langle D, \Theta_D \rangle \right) \tag{4.41}$$

where $P_{\mathbf{O}}(T)$ is the joint probability distribution defined by T over the variables $\mathbf{O}$.

Definition 53. Two latent structures, L and L' are **equivalent**, written as $L' \equiv L$ *iff*

$$L \preceq L' \wedge L' \preceq L \tag{4.42}$$

Definition 54. A latent structure L is **minimal** with respect to a class, $\mathcal{L}$, of latent structures *iff*

$$\forall L' \in \mathcal{L}, L' \preceq L \Rightarrow L \equiv L' \tag{4.43}$$

Definition 55. $L = \langle D, \mathbf{O} \rangle$ is **consistent** with a distribution $\hat{P}$ over $\mathbf{O}$ if D can accommodate some theory that generates $\hat{P}$, that is:

$$\exists \Theta_D \; : P_{\mathbf{O}} \left(\langle D, \Theta_D \rangle \right) = \hat{P} \tag{4.44}$$

Inferred causation is then defined as follows:

Definition 56. Given $\hat{P}$, a variable C has a **causal influence** on E *iff* there exists a directed path $C \mapsto E$ in every minimal latent structure consistent with $\hat{P}$.

While the definition given so far would allow for the discovery of some causal, as defined, relationships the possible search space is still very large as its possible for vanishing dependencies

to be a result of the parameters not of the structure. To avoid this problem Pearl and Verma add a new requirement stability[5]. It has been shown, under various different conditions, that with an underlying smooth distribution the chance of randomly selecting a distribution that is not stable is measure zero [Meek 1995, Spirtes et al 1993]. Let $I(P)$ denote the set of all conditional independence relationships embodied in P.

Definition 57. A causal theory $T = \langle D, \Theta_D \rangle$ generates a **stable** distribution *iff* it contains no extraneous independencies, that is,

$$I(P(\langle D, \Theta_D \rangle)) \subseteq I\left(P\left(\left\langle D, \Theta'_D \right\rangle\right)\right) \tag{4.45}$$

for any set of parameters Θ_D.

One drawback of this condition is that it excludes deterministic relationships. Assuming no hidden variables, that is, when $\mathbf{U} = \mathbf{O}$ two causal models are equivalent *iff* their DAGs have the same links and the same set of uncoupled head-to-head nodes. An uncoupled head-to-head node has the form $a \rightarrow c \leftarrow b$ where a and b are not directly linked[6]. Verma and Pearl [Verma and Pearl 1990] identified the maximal set of sound constraints imposed by a latent structure on a distribution which allow the recovery of sound fragments of latent structures.

Definition 58. A latent structure $L_{\mathbf{O}} = \langle D_{\mathbf{O}}, \mathbf{O} \rangle$ is a **projection** of another latent structure L *iff* every unobservable variable of $D_{\mathbf{O}}$ is a parentless common cause of exactly two non-adjacent observable variables, and for every stable distribution P generated by L, there exists a stable distribution P' generated by $L_{\mathbf{O}}$ such that $I(P_{\mathbf{O}}) = I\left(P'_{\mathbf{O}}\right)$.

This leads to the theorem that any latent structure has at least one projection.

Definition 59. For any latent structure L, ***core(L)*** is defined as the hybrid graph satisfying: two nodes are adjacent in $core(L)$ *iff* they are adjacent or they have a common unobserved cause in every projection of L; a link between a and b has an arrowhead pointing at b *iff* $a \rightarrow b$ or a and b have a common unobserved cause in every projection of L.

For any latent structure $L = \langle D, \mathbf{O} \rangle$ and an associated theory $T = \langle D, \Theta_D \rangle$ if $P(T)$ is stable then every arrowhead identified by IC is also in $core(L)$. So, if every link of the directed path $C \rightarrow E$

[5] This is similar to faithfulness, see definition 41 in section 4.1.

[6] This is often called a v-structure, see definition 43 section 4.1.3.

is marked in $core\left(\hat{P}\right)$, that is, denoted with the $\mapsto$ arrow, then C has a causal influence on E according to $\hat{P}$. Pearl and Verma then define three types of relationship:

> **Definition 60. Potential Cause**: A variable X has a potential causal influence on another variable Y (inferable from $\hat{P}$) if: X and Y are dependent in every context; there exists a variable Z and a context S such that (i) X and Z are independent given S and (ii) Z and Y are dependent given S.

> **Definition 61. Genuine Cause**: A variable X has a genuine causal influence on another variable Y if there exists a variable Z such that: X and Y are dependent in any context and there exists a context S satisfying (i) Z is a potential cause of X (ii) Z and Y are dependent given S (iii) Z and Y are independent given $S \cup X$.

> **Definition 62. Spurious Association**: Two variables X and Y are spuriously associated if they are dependent in some context S and there exists two other variables Z_1 and Z_2 such that: Z_1 and X are dependent given S; Z_2 and Y are dependent given S; Z_1 and Y are independent given S; Z_2 and X are independent given S.

These different types of relationship form the basis of the IC algorithm.

The IC algorithm is a powerful tool for finding causal relationships in data, its assumptions largely amount to the distribution of the data system being stable, that is, a distribution which could be generated by a BN. The IC algorithm starts with a graph with no links and adds undirected links when it finds variables which are related in every context. Even without additional information some ordering of the direction of causal influence can be achieved. Unfortunately, as with the purely Bayesian methods, in general, the performance of the IC algorithm will degrade exponentially with an increasing number of variables, although for sparse graphs it can still perform acceptably [Verma and Pearl 1990, Spirtes & Glymour 1990]. The IC algorithm is quite simple and has been enhanced in a number of ways to improve performance or add functionality. The IC algorithm as shown uses no prior knowledge to augment either link discovery or determining the direction of a link. Node ordering using either timing information or prior knowledge can easily be used to augment determination of the direction of influence as shown in [Zhang, Baral & Kim 2005]. Another issue is the problem of deterministic relationships. Wei

Luo [Luo 2006] derives a modified version of IC that can cope with known deterministic relationships, where such relationships are not initially known the data can be checked to determine the likely existence of deterministic relationships prior to using the modified IC algorithm.

4.3.2 The FCI Algorithm

The Fast Causal Inference, FCI, algorithm is another graph based learning algorithm. Like IC it uses conditional independence relationships to define links in the graph. However, FCI imposes stricter constraints on the type of relationships its guaranteed to work with, linear relationships[7] with a Normal distribution are supported, although it has proved to be quite successful on a number of non-linear problems. The FCI algorithm is shown in algorithm 4.5. Like IC, FCI assumes the data to be generated by some BN. Thus it assumes the structure it is looking for conforms to the Causal Markov condition and that the data is faithful (stable) to some DAG, that is, the independencies in the data match those in the DAG. The algorithm uses d-separation as a test for causal (in)dependence. D-separation [Pearl 1994] defines the type of conditional (in)dependence relationships entailed by the Markov condition. We need to define some terms to help with the algorithm.

Definition 63. A **collider** is a node with multiple parents.

Definition 64. $Adjacencies(D,A)$ is the set of variables (nodes) in the graph D adjacent to the variable A, that is the variables with a direct link to A.

Definition 65. $Possible-D-SEP(A,B)$ is defined for a partially oriented inducing path graph, π, as follows: If $A \neq B$, V is in $Possible-D-SEP(A,B)$ *iff* $V \neq A$, and there is an undirected path U between A and V in π such that for every sub-path $\langle X,Y,Z\rangle$ of U either Y is a collider on the sub-path, or Y is not a definite non-collider on U, and X, Y and Z form a triangle in π.

In terms of link connections Sprites, Glymour and Scheines identify three types of termination related to the direction of causal influence. A small circle indicates the direction of influence is uncertain as in $A \circ -$, a straight tail indicates causal influence from the variable as in $A-$,

[7] Strictly it is structural equation models.

and an arrow indicates the direction of influence is towards the variable as in $A \leftarrow$, they also define a don't care matching operator which will match any of the other link endings denoted by $A * -$. FCI is, in effect, a development of the Causal Inference, CI [Spirtes et al 1993], and PC [Spirtes et al 1991] algorithms. The PC algorithm is similar, computationally more expensive and requires that there are no hidden variables. While the CI algorithm produces a better partly oriented inducing path graph than FCI it is computationally intractable for large networks. However, where as the CI algorithm can often cope with latent variables causing causal insufficiency [Klopotek 1994], the same cannot be said for FCI [Klopotek 2000]. FCI starts with a complete graph, unlike IC which starts with an empty one, and deletes links where conditional independence tests show they are not required. Like IC determining link direction starts with colliders and works to ensure consistency with no directed cycles.

Algorithm 4.5 The FCI Algorithm

1. Form the complete undirected graph D on the variables $\mathbf{O}$
2. $n = 0$
   (a) repeat
      i. repeat
         A. select an ordered pair of variables X and Y that are adjacent in D such that $Adjacencies(D,X) \setminus \{Y\}$ has cardinality greater than or equal to n, and a subset S of $Adjacencies(D,X) \setminus \{Y\}$ of cardinality n, and if X and Y are d-separated given S [this is equivalent to $(X;S;Y)$ in the IC notation] delete the link between X and Y from D, and record S in $Sepset(X,Y)$
      ii. until all ordered variable pairs of adjacent variables X and Y such that $Adjacencies(D,X) \setminus \{Y\}$ has cardinality greater than or equal to n and all subsets S of $Adjacencies(D,X) \setminus \{Y\}$ of cardinality n have been tested for d-separation
      iii. n=n+1
   (b) until for each ordered pair of adjacent variables X, Y, $Adjacencies(D,X) \setminus \{Y\}$ is of cardinality less than n
3. Let D' be the undirected graph resulting from step 2. Orient each link as $\circ - \circ$. For each triple of variables A,B,C such that the pair A,B and the pair B,C are each adjacent in D' but the pair A,C are not adjacent in D', orient $A * - * B * - * C$ as $A * \rightarrow B \leftarrow * C$ *iff* B is not in $Sepset(A,C)$
4. For each pair of variables A and B adjacent in D', if A and B are d-separated given any subset S of $Possible - D - SEP(A,B) \setminus \{A,B\}$ or any sunset S of $Possible - D - SEP(B,A) \setminus \{A,B\}$ in D remove the link between A and B, and record S in $Sepset(A,B)$ and $Sepset(B,A)$

4.3.3 The LCD Algorithm

The paper which introduces the Local Causal Discovery, LCD, algorithm [Cooper 1997] seems to be mostly intended as an introduction to constraint-based learners, as such Cooper makes a clear outline of the assumptions, mathematics and, at least some of, the limitations of this type of algorithm. LCD requires seven assumptions, the first six of which apply to almost all causal discovery algorithms, they are:

Database completeness Let D be a database of cases such that each case contains a value for each variable in set **V**.

There are a number of methods for dealing with missing values that can be used to remove this requirement: entries with missing values can be ignored; 'appropriate' values can be inserted; a special value, such as, *missing* could be used where no value is supplied. Each of these options has benefits and costs and ultimately its simpler to just require a complete database. There are two immediate reasons why simply ignoring entries with missing values can cause problems. Firstly all the statistical tests used by the constraint-based causal learners are correct in the large limit, that is, when supplied with a lot of data. Real world databases may not be very large and removing entries with missing values could lead to the tests becoming unreliable. Secondly the missing values may not be random, but the very fact of a missing value might indicate something about the state of the system at that time. Thus simply ignoring missing items could lead to selection bias, see Selection Bias below.

Discrete variables Each variable in **V** has a finite, discrete number of possible values.

This assumption is only for convenience, continuous variables can be discretised into an arbitrary number of possible value ranges if required. Discrete variables with a finite number of values simply make the calculations easier.

A Bayesian Network Causal Model The underlying causal processes that exist among the variables in **V** can be modelled using some Bayesian Network G, which might contain hidden variables not in **V**.

A BN imposes a number of conditions. The network model will be a DAG, and the Markov condition applies to the network. Hence the Causal Markov condition applies to the relationships that can be represented and therefore discovered. In the BN we define S to be the graphical structure of G and P is the joint probability distribution represented by G.

Causal Faithfulness condition For all disjoint sets **A**, **B** and **C** in **V**, if in S we have that **A** is not d-separated from **B** by **C**, then in P we have that **A** and **B** are conditionally dependent given **C**.

This requirement is necessary if we assume that the underlying system, whose relationships we are trying to discover, can be modelled by a BN. This is identical to the stability requirement of the IC algorithm.

No Selection Bias If $\mathbf{V}'$ denotes an arbitrary instantiation of all the variables in **V**, then $\mathbf{V}'$ is sampled for inclusion in D with probability $P(\mathbf{V}' \mid G)$ where G is the causal Bayesian network to be discovered.

This just states that the database is a fair random sample of the underlying distribution. If this is untrue it may not be possible to discover the underlying distribution. It may be possible to detect when selection bias exists [Spirtes et al 1995].

Valid Statistical testing If T is the test used to determine conditional independence, such as the chi-squared test, consider the sets of variables **A**, **B** and **C** in **V**. If in P we have that **A** and **B** are conditionally independent given **C**, then **A** and **B** are conditionally independent given **C** according to test T applied to the data in D.

Since we know that the distribution within D is a fair representation of the distribution in P (no selection bias) we need our statistical test T to be able to uncover the relationships within the database.

The above six assumptions are common to most constraint-based causal learners, and BN-based causal learners in general. The next assumption is, as far as we are aware, unique to LCD, but its really just a simplification to allow clarity in the explanation of causal discovery.

A known uncaused entity There is a designated variable W in **V** that is not caused by any other variable in **V**.

The above assumptions allow for easy causal discovery in the situation where we have the statistical tests $Dependent(A, B \mid C)$ that returns true *iff* variable A is dependent with variable B conditional on variable C, where C might be empty, that is $\emptyset$, and similarly $Independent(A, B \mid C)$ which return true *iff* A is independent of B conditional on C, then when given:

$$Dependent(W, X \mid \emptyset) \wedge Dependent(X, Y \mid \emptyset) \wedge Independent(W, Y \mid X) \tag{4.46}$$

we can conclude that X causes Y.

The LCD algorithm is shown in algorithm 4.6. This is a very simple algorithm and one of its great advantages is that is has a time complexity of $O\left(n^2 m\right)$ where there are n variables each of which has m cases in the database. This is due, in part, to the simple nature of the conditional tests used. This has an added benefit in that low order conditional tests tend to be more reliable than high order tests. However, even within the restricted set of cases where it can potentially determine the direction of causality, it will fail to detect many such cases. LCD does not produce a complete network, rather as written it simply produces pairwise assertions of causal effect. This is not a major problem as post processing could produce at least a partial network from its output. Cooper suggests LCD could be made more resilient to errors in the test $Independent\,(W,Y \mid X)$ by adding a fourth test:

$$Dependent\,(W,X \mid \emptyset) \wedge Dependent\,(X,Y \mid \emptyset) \wedge Independent\,(W,Y \mid X) \wedge Dependent\,(W,Y \mid \emptyset) \tag{4.47}$$

However, Cooper suggests it is also possible do away with the requirement of an uncaused variable W, instead we can still perform causal discovery, concluding X causes Y when we can find two variables $W1$ and $W2$ such that:

$$\begin{aligned} Dependent\,(W1,X \mid \emptyset) \quad &\wedge \quad Dependent\,(W2,X \mid \emptyset) \wedge Independent\,(W1,W2 \mid \emptyset) \\ &\wedge \quad Dependent\,(X,Y \mid \emptyset) \wedge Independent\,(W1,Y \mid X) \end{aligned} \tag{4.48}$$

A reasonably through analysis of this type of 'Y' structure and, under given circumstances, a proof of the correctness of the causal assumption 'X causes Y' can be found in the paper [Mani, Spirtes & Cooper 2006].

Silverstein et al [Silverstein et al 2000] examine the statistical tests used in LCD and suggest a two tailed variant using the chi-squared test and an indication of the proportion of data items in which a particular pattern occurs. In addition to the basic causality test defined in the LCD algorithm, which they refer to as a CCC test, they add what the refer to as the CCU test. The CCU test can be defined as:

$$\begin{aligned} Dependent\,(W1,X \mid \emptyset) \quad &\wedge \quad Dependent\,(W2,X \mid \emptyset) \wedge Independent\,(W1,W2 \mid \emptyset) \\ &\wedge \quad Dependent\,(W1,W2 \mid X) \end{aligned} \tag{4.49}$$

from this you can conclude $W1 \rightarrow X$ and $W2 \rightarrow X$[8].

Algorithm 4.6 The LCD Algorithm

1. **procedure** $LCD(W, \mathbf{V}, D)$;
2. {Input: A variable W , which is assumed not to be caused by any other variable in the set $\mathbf{V}$ of discrete, measured variables in complete database D.}
3. {Output: A printout of causal relationships that are discovered.}
4. **for** $X \in \mathbf{V} \backslash \{W\}$ **do**
   (a) **if** $Dependent(W, X \mid \emptyset)$ **then**
   (b) **for** $Y \in \mathbf{V} \backslash \{W, X\}$ **do**
      i. **if** $Dependent(X, Y \mid \emptyset)$ **then**
         A. **if** $Independent(W, Y \mid X)$ **then**
         B. **write**('the data supports ', X , ' as a cause of ', Y);
   (c) **end** {for};
5. **end** {for};
6. **end** {LCD}.

4.4 Summary

Attempting to learn causal networks from data is a hard problem. Most learners assume the generating network is a BN and so they are trying to learn a DAG. It is known that even without attempting a causal analysis this is an NP-Hard problem. However, assuming the target is a BN does provide some useful information. In the context of a BN it is known that the **Markov** condition holds and this has been extended to the **Causal Markov** condition which can be used to help resolve causal ambiguities. Many learners make the assumption that the data is **faithful** to the BN which helps determine the required structure, and has strong implications for how the (in)dependence relationships within the data determine the possible BN structure. There is also the issue of variables that are direct causes of variables in the data, but are not themselves contained within the data, such hidden, sometimes called latent, variables can cause problems both in determining the correct structure and in determining the causal relationships within that structure.

The Bayes' rule based learners often deal best with the various difficulties posed by the causal learning task, and potentially could provide a 'best fit' network. The problem with pure

[8]This is only assumed to be correct in the absence of hidden and confounding variables.

Bayes rule based learners is that the computational complexity of the learning task tends to be super exponential in terms of the number of variables considered. This makes such techniques impractical for many real world learning tasks. Variations on Bayes rule learners which use a heuristic search exist and non Bayes rule whole network learners overcome some of the problems, but like many search based learners they can suffer from problems of local maxima. While reducing the search space by pre-filtering possible solutions helps, it risks removing the 'best fit' network from those considered. The basic problem with whole network learners is that the search space of possible networks is so large as to be intractable. This means assumptions about the target network have to be made and heuristic methods used to limit the possible networks.

Constraint-based learners have the advantage that, by considering only a relatively small subset of the variables at any point, they have a computationally simpler task. However, it can be difficult to know what structure at a local level gives rise to the overall 'best fit' network. Constraint-based learners also require a significant amount of data if their statistical tests are to have the required *large sample* properties[9]. Latent variables can be a particular problem for constraint-based learners as focusing on only a small number of variables, at any one time, can potentially magnify their effect. Despite these difficulties useful learning algorithms have been developed and it is, as this thesis attests, an area of current research.

The LUMIN learner introduced in the next chapter uses a number of the techniques that appear in other causal learners. The output is a graphical network whose nodes represent the variables involved and whose links denote causal influence, however, unlike most learners the output is not a DAG as it allows for both undirected and links and cycles in the dependencies. LUMIN builds the network starting from a graph with no links like the IC algorithm. The LUMIN learner uses mutual information in a similar fashion to Chow and Liu, and Lam and Bacchus, that is, it is used to measure the strength of relationship between two variables. It also uses the conditional version of the same test to help determine the direction of causal influence. Most constraint-based causal learners look for colliders to help determine the direction of causal relationships, LUMIN in particular examines related triplets of variables to both find colliders and help determine the direction of causal influence for non-colliders. Some causal learners use a node ordering to aid with determining the direction of causal influence. **K2,** for example, orders all the nodes and hence predetermines the direction of causality for every possible relationship. LUMIN can also

[9]The large sample property is based on the law of large numbers and central limit theorem and derives their asymptotic properties as sample size, n, goes to infinity.

use a form of predetermined causal influence through the use of rankings within a domain see section 5.3.4. The predetermination of causal influence used by LUMIN is generally considerably more flexible than those used by most other causal learners. A number of learners use heuristic tests to help reduce the complexity of the learning task, LUMIN also uses heuristic tests to help simplify chains of causal influence see section 5.4.

Chapter 5

The LUMIN Learner

This chapter introduces the LUMIN (Learning Using Mutual INformation) program. The LUMIN learner and the analysis behind it form the major innovation of this thesis. This chapter starts by discussing the motivation for creating a new causal learner. It examines some of the limitations of current causal learners. Section 5.2 examines some of the properties we considered desirable for the new learner. We then examine the algorithm itself, in section 5.3, noting the assumptions that are required for this type of causal discovery, and at the expected behaviour of causal relationships under those assumptions. The statistical tests used are explained and their relationship examined. The statistical tests used by the LUMIN learner do not produce a perfect graph. In addition to the common problem of overfitting there are specific issues that arise with this type of causal discovery. Section 5.4 discusses some of these and explains the steps that are taken to reduce their effect. One of the key issues with many existing causal learners is that of computational complexity. Many causal learners are unable to cope with large datasets, the computational task becomes intractable. Section 5.5 examines the computational complexity of the LUMIN learner. Sections 5.6 and 5.7 examine some limitations of the LUMIN learner. Lastly section 5.8 gives a summary of the chapter.

5.1 Motivation

The learning of causal relationships from data is one of the key steps to understanding how the system that generated the data works. This is one of the cornerstones of science. In many situations it is not possible to do more than observe a system, so causal relationships can only

be learnt through observation. The motivation for the LUMIN program was to provide a tool suitable to help with identifying causal relationships from such data.

There are already in existence a number of algorithms which attempt to learn causal relationships from data, do a pretty good job of it, and a number produce a functional Bayesian network as output. Why develop a new one? Many existing methods of discovering causal relationships in data are based around BNs and as such impose restrictions on the possible forms of the causal relationships. Standard BNs require an acyclic graph, while it appears that many natural systems incorporate some feedback[1].

Example 1 Many animals work at maintaining their temperature within a given range, and incorporate specific feedback mechanisms to this end.

Example 2 The atmospheric concentration of carbon dioxide, CO_2. The current belief is that increased concentrations of CO_2 lead to higher average global temperatures, and the increased temperatures lead to the release of more CO_2 both from previously stable sources, like deposits trapped in permafrost and through a reduction in CO_2 uptake in plants.

A crucial point here is that causal cycles can exist, but the feedback occurs after the conditions which caused it. Thus, when taking a snapshot in time of the system there would be no obvious feedback taking place. However, since learning usually involves taking a number of data points spread out over time, it is possible these feedback loops are represented in the data.

Another issue is that of computational complexity. The majority of causal discovery programs produce a DAG. In general learning the structure of a DAG is an NP-hard problem [Chickering et al 1994]. Thus, one aim was to produce a system which could be used on *real world* datasets and produce results in polynomial time.

5.2 Design Considerations

In choosing an algorithm for causal discovery we wanted to try to avoid some of the limitations commonly found in causal discovery systems. There are a number of design considerations with any program. In particular we wanted to address potential weaknesses with some other causal learning techniques.

[1] Dynamic BNs, [Kjærulff 1992, Ghahramani 1998], can represent networks which include feedback, but we know of no method to learn arbitrary dynamic BNs.

5.2.1 Causal Loops

We wanted to be able to discover causal loops. This immediately breaks the Markov condition and means that potentially the output of the program will not be in a suitable form for a standard BN. The algorithm can, in principle, discover causal loops. However, the current version will not find direct causal loops, those involving only two variables, of the form shown in figure 5.1, unless a time series of data is available. It may be able to find indirect causal loops, those with

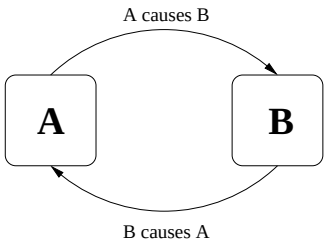


Figure 5.1: Direct Causal Loop

more than two variables, such as the one shown in figure 5.2. The discovery of a causal loop

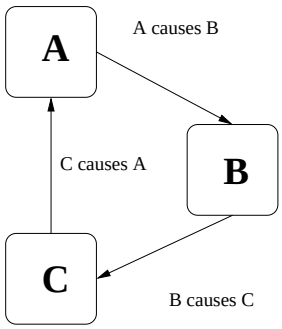


Figure 5.2: Indirect Causal Loop

implies that the data records contain values separated by sufficient time to allow the feedback to be detected. Thus any causal loops found in *raw* data should be treated with caution.

5.2.2 Non-Linear Relationships

The idea of extracting causal relationships from data is not new. It has been common for some time in the social sciences and elsewhere to use regression models, often linear, to build causal models. There is no widespread agreement that this is a reasonable thing to do as illustrated by discussions between Keynes and Tinbergen [Keynes 1939, Tinbergen 1940, Keynes 1940]. The assumption of linear relationships is still common [Spirtes et al 1993], either because it reduces the computational complexity, or because of a genuine belief that the majority of *real world* causal relationships are linear.

We decided to use Mutual Information, MI, as a metric to look for relationships between variables. MI has the advantage that it should be able to detect a variety of both linear and non-linear relationships. MI measures the amount of information that is shared between two variables. For two discrete variables X and Y, their mutual information $I(X;Y)$ is shown in definition 6, section 2.3. This shows MI can be expressed in terms of probabilities, equation 2.6, and also in terms of entropy as shown in equation 2.7. MI is symmetric, that is

$$I(X;Y) = I(Y;X)$$

and

$$I(X;Y) = 0 \textit{ iff } X \perp\!\!\!\perp Y$$

also

$$I(X;X) = H(X)$$

as

$$H(X|X) = 0$$

Since $H(X) \geq H(X|Y)$ it is clear that $I(X;Y) \geq 0$, and MI is bounded such that:

$$0 \leq I(X;Y) \leq \min(H(X), H(Y)) \tag{5.1}$$

Definition 66. We can define a normalised version of MI, **NMI**, such that:

$$NMI(X,Y) := \frac{I(X;Y)}{\min(H(X), H(Y))} \tag{5.2}$$

hence

$$0 \leq NMI(X,Y) \leq 1 \tag{5.3}$$

Since NMI has a fixed range and increases with increasing mutual information between the variables it allows us to make direct comparisons of the mutual information between different pairs of variables.

5.2.3 Use of Domain Knowledge/Clarity of Ignorance

When analysing the behaviour of a system we will have some knowledge about it, ranging from a complete understanding of all the parts and their interactions, to total ignorance. We wanted to be able to represent and use whatever causal knowledge was available to the learner. To this end any likely causal relationships or known temporal relationships can be indicated as follows:

The Direction of Causality the direction of a possible relationship can be indicated by means of a ranking within a given domain[2]. Rankings are organised in a pseudo time-line from low values to high values. A low ranking variable can be a cause of a higher ranking variable, but not visa versa. If two variables have the same ranking (in all shared domains) then either might be a cause of the other. The LUMIN program allows a number of domains with a range of rankings[3] and a variable can belong to any number of domains. All variables belong to the 'General' domain with a default equal ranking[4]. If the rankings in any shared domain forbid a particular causal relationship then the relationship is forbidden[5]. If more information is known then it can be used, by either changing the rankings of variables in the 'General' domain or by adding further domains.

Known Causal Relationships if existing causal relationships within the data are already known then this information can be supplied and the relationship will be added to the output graph. The information supplied is the existence of the link and, if known, its direction. This feature is useful not only in adding what is already known, but to test *what if* scenarios as the information is used when determining the remaining causal relationships within the data.

Forbidden Causal Relationships if it is known that two variables have no causal relationship this can be indicated. Similar to defining the existence of a causal relationship this forbids the creation of a causal link between two variables.

5.2.4 Dealing with Incomplete Data

It is not uncommon to find a number of values missing in some records in a dataset. We wanted to be able to make the best use of all the available data with as few assumptions about the dataset and

[2]Domains and rankings are defined in section 5.3.4.

[3]The range of values a ranking can take and the maximum number of rankings are determined by the implementation of `unsigned int` and `vector` in the C++ compiler.

[4]The default value is 10.

[5]This refers to the direction of causality not the existence of a causal relationship.

its relationships as possible. If a record had missing values we still wanted to be able to use the information contained in the record. Various methods for dealing with missing data have been devised, including the use of the EM algorithm [Dempster et al 1977], Markov Chain Monte Carlo methods [Chickering and Heckerman 1996], Bayesian methods [Cooper & Herskovits 1992] and Bound and Collapse [Ramoni and Sebastiani 1997]. However, each of these methods imposes some restriction on or makes additional assumptions about the underlying dataset.

The LUMIN program makes no attempt to fill in any missing data items. LUMIN examines subsets of the attributes in the dataset and all records which contain values for the variables being examined will be used. Thus a record with missing values will be used when analysing a set of variables for which is has values. There are possible problems with this approach; if the available data is limited, by not using some method of substituting for the missing values there may be too little data for the statistical tests to be valid; it is also possible that missing data may indicate a special state of the system, so ignoring those entries with missing data may introduce a selection bias.

5.2.5 Computational Cost

It is possible to perform a complete and sound analysis of a dataset with a few initial assumptions [Pearl and Verma 1991, Spirtes et al 1993, Heckerman et al 1995]. However, such analysis tends to be computationally expensive and to perform in worse than polynomial time with respect to the number of variables and, in some cases, the number of data records. We were looking for an algorithm that would be suitable for analysing large datasets, so what was required was something with a polynomial time performance. The LCD algorithm, [Cooper 1997], provides limited causal discovery in polynomial time. However, as it seems to be mostly an introduction to local causal discovery, it is rather limited in that it does not determine the direction of discovered relationships except in the case where one variable is known to have no cause within the dataset. [Silverstein et al 2000] both extends causal relationship discovery in LCD by adding a test for unshielded colliders, and highlights that the LCD algorithm is sensitive to small errors in its dependence and independence tests, since each error can potentially propagate causing changes in the causal relationships discovered. Some of the extensions to the LCD algorithm in [Silverstein et al 2000], in particular those to which reduce erroneous discovery, rely on the binary nature of the data examined in the paper, and so are of limited usefulness.

5.2.6 Simple Input Data Format

In order to allow easy use of the system by others, and to simplify testing with a variety of existing datasets we decided LUMIN should use an extended version of the C4.5 data format [Quinlan 1993]. The basic C4.5 data format works, but additional files can be used to specify domain and ranking information and the presence or absence of specific relationships.

5.2.7 Simple Output Format

The output of LUMIN is a graph structure with directed, and potentially, undirected links. We decided to use the DOT[6] graph description language for the output. DOT is a simple, plain text based language with an associated set of programs that allow the display of the graph on a number of different computer systems. In the output graph variables are indicated by circular nodes and the relationships between them are indicated by lines joining the nodes. Where a line meets a node it has one of three possible endings, a small circle, a straight tail, or an arrowhead. The small circle indicates no conclusion could be reached about the direction of causality with the adjacent node, a tail indicates that the direction of causal influence is from the adjacent node, and an arrowhead indicates the direction of causal influence is into the adjacent node. This leads to the cases shown in figure 5.3. In each example in the figure there appears to be a relationship

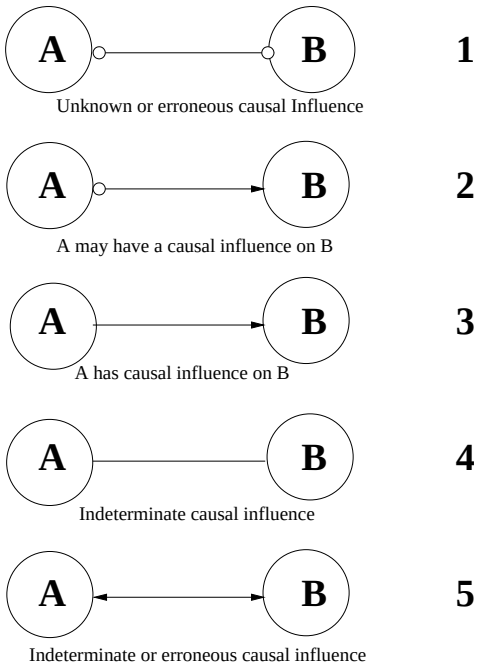


Figure 5.3: Output Graph Relationships

[6]See `http://www.graphviz.org/doc/info/lang.html` for the full specification.

between variables **A** and **B** in addition in example:

1. nothing could be determined about the direction of causal influence
2. there appears to be a causal influence into **B**, but its not certain its from **A**
3. there appears to be a causal influence from **A** to **B**
4. there appears to be a causal relationship between **A** and **B**, but is it not possible to determine the causal direction
5. there appears to be causal influence in both directions between **A** and **B**. This may indicate the presence of hidden or confounding variables, or that feedback is occurring.

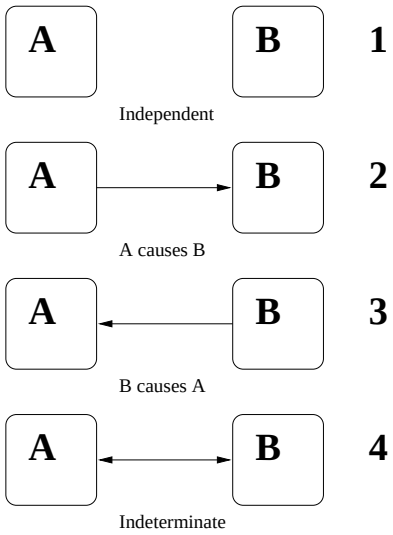


Figure 5.4: Causal Relationships with 2 Variables

5.3 The Algorithm

This section describes the basic algorithm used by the LUMIN program. When dealing with the values of the variables representing the measurements of attributes of the system under study we will just refer to them as the value of a variable or as a variable for brevity.

5.3.1 Assumptions

All learners have to make assumptions to make the learning process possible. In general making few assumptions allows for a greater flexibility in the learner, but requires greater computational effort. The LUMIN learner makes, at least, the following assumptions:

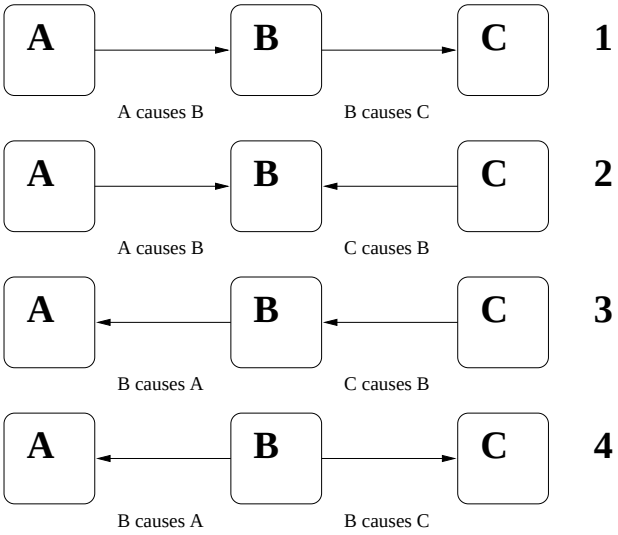


Figure 5.5: *Interesting* Causal Relationships with 3 Variables

Assumption 1 The system under investigation is ultimately deterministic.

Individual parts of the system have deterministic rules governing all their properties. So if the value of some variable within the system is represented by the variable Y and its causes, that is, those variables which have a direct causal influence on its value, are represented by the variables $X_1,\ldots,X_m$ then there will be some deterministic function f_y such that

$$Y = f_y(x_1,\ldots,x_m) \tag{5.4}$$

where $x_1 \in X_1,\ldots,x_m \in X_m$. This does not rule out the discovery of more complex or even some aggregate relationships[7], such as that between pressure and temperature for a gas, or diffusion for a liquid, but it is a philosophical starting point for the causal analysis. Using the common BN terms we would say that the variables $X_1,\ldots,X_m$ are the parents of Y.

Assumption 2 If $X_1,\ldots,X_m$ are the parents of Y then each X_i will share some mutual information with Y. This is illustrated with a single parent in case 2 in figure 5.4, where variable A is a cause of variable B.

Assuming we keep the values of $x_2,\ldots,x_m$ constant we effectively have the single parent case $Y = f_y(X_1)$ so we see that from equation 2.6 we have

$$I(X_1;Y) = \sum_{y\in\mathbf{Y}}\sum_{x_1\in\mathbf{X}_1} p(x_1,y)\log_2\left(\frac{p(x_1,y)}{p(x_1)p(y)}\right) \tag{5.5}$$

[7]However, there are well known statistical problems with some aggregates, such as Simpon's paradox [Simpson 1951].

where $\mathbf{Y}$ and $\mathbf{X}_1$are the set of values of y and x_1 respectively. Now we know that

$$p(x_1,y) = \begin{cases} y \neq f_y(x_1) & 0 \\ y = f_y(x_1) & \frac{1}{\|\mathbf{X}_1\|} \end{cases} \tag{5.6}$$

so we have

$$\begin{aligned} I(X_1;Y) &= \sum_{x_1\in\mathbf{X}_1}\sum_{y\in\mathbf{Y},y\neq f_y(x_1)} p(x_1,y)\log_2\left(\frac{p(x_1,y)}{p(x_1)p(y)}\right) + \sum_{x_1\in\mathbf{X}_1,y=f_y(x_1)} p(x_1,y)\log_2\left(\frac{p(x_1,y)}{p(x_1)p(y)}\right) \\ &= \sum_{x_1\in\mathbf{X}_1}\sum_{y\in\mathbf{Y},y\neq f_y(x_1)} 0\log_2\left(\frac{p(x_1,y)}{p(x_1)p(y)}\right) + \sum_{x_1\in\mathbf{X}_1,y=f_y(x_1)} \frac{1}{\|\mathbf{X}_1\|}\log_2\left(\frac{\frac{1}{\|\mathbf{X}_1\|}}{\frac{1}{\|\mathbf{X}_1\|}\frac{1}{\|\mathbf{X}_1\|}}\right) \\ &= 0+\log_2(\|\mathbf{X}_1\|) \end{aligned} \tag{5.7}$$

this is making the additional simplifying assumptions that $f_y(x_1 = a) = f_y(x_1 = b) \Rightarrow a = b$ and that we can take $0\log_2(0) = 0$. Now the entropy of $\mathbf{X}_1$, $H(\mathbf{X}_1)$ is

$$\begin{aligned} H(\mathbf{X}_1) &= -\sum_i^{\|\mathbf{X}_1\|} p(x_{1_i})\log_2(p(x_{1_i})) \\ &= -\sum_i^{\|\mathbf{X}_1\|} \frac{1}{\|\mathbf{X}_1\|}\log_2\left(\frac{1}{\|\mathbf{X}_1\|}\right) \\ &= \log_2(\|\mathbf{X}_1\|) \end{aligned} \tag{5.8}$$

where x_{1_i} is the ith value of x_1 from $\mathbf{X}_1$. Thus given that $H(\mathbf{X}_1) = H(\mathbf{Y})$ we have

$$\begin{aligned} NMI(X_1,Y) &= \frac{I(X_1;Y)}{H(X_1)} \\ &= \frac{\log_2(\|\mathbf{X}_1\|)}{\log_2(\|\mathbf{X}_1\|)} \\ &= 1 \end{aligned} \tag{5.9}$$

which is what would be expected of a deterministic relationship. In this example we assumed that the values of $x_2,\ldots,x_m$ were fixed, however, even if we allow them to vary, it should still be true in the large sample limit, that the larger the value of $NMI(X,Y)$ the greater the likelihood of there being a causal relationship between X and Y. One problem here is the usual statistical problem of soft tests, there is no value, other than zero, which allows you to be sure there is no causal relationship. In practise a value will need to be chosen to indicate the presence or absence of a likely causal relationship. Definitions of cause and effect often refer to interventions on the

cause related item/attribute changing the effect related item/attribute, in this context the change in the value of a parent is the virtual intervention. The intervention is virtual since we do not change the parent, but instead rely on it having multiple values in the data which allow us to examine the effect of the parent having different values. Since MI is a statistical measure it is possible that any two variables will share some amount of information by chance. The assumption we make is that causally related variables will share a significant amount of MI.

Assumption 3 Conditional Mutual Information and its normalised variant are comparable to mutual information and its normalised variant, and can be used to indicate possible causal relationships.

Conditional Mutual Information, CMI, measures the mutual information shared between two variables when conditioned on a third. In effect what it does is to find out how much information is shared between two variables when they are sampled with the conditioning variable having a fixed value. This is computed over the range of possible values of the conditioning variable. In a sense CMI is a bit like a limited type of d-separation, in that it looks at the shared information between two variables when communication via a third variable, the conditioning variable, is forbidden.

Definition 67. Conditional Mutual Information is defined as follows, to find the mutual information between variables X and Y conditioned on variable Z, $I(X;Y \mid Z)$ we have

$$I(X;Y \mid Z) := \sum_{z \in \mathbf{Z}} \sum_{y \in \mathbf{Y}} \sum_{x \in \mathbf{X}} p(x,y,z) \log_2 \left(\frac{p(z)\, p(x,y,z)}{p(x,z)\, p(y,z)} \right) \tag{5.10}$$

where $\mathbf{X}$, $\mathbf{Y}$, and $\mathbf{Z}$ are sets of the values of the variables X, Y and Z respectively.

As with MI, $I(X;Y \mid Z) \geq 0$ and is symmetric in X and Y. This can be written in terms of entropy to give

$$I(X;Y \mid Z) = H(X,Z) + H(Y,Z) - H(X,Y,Z) - H(Z) \tag{5.11}$$

this can be normalised to the $[0,1]$ range.

Definition 68. Normalised Conditional Mutual Information, NCMI, is defined as

$$NCMI(X,Y \mid Z) = \frac{I(X;Y \mid Z)}{\min(H(X,Z), H(Y,Z)) - H(Z)} \tag{5.12}$$

LUMIN uses the change in normalised MI caused by conditioning as a test for causal influence so its important to understand the relationship between NMI and NCMI. If we assume X and Y are both independent of Z we can see that $NMI(X,Y)$ compares with $NCMI(X,Y \mid Z)$ as follows

$$\begin{aligned} I(X;Y \mid Z) &= \sum_{z\in\mathbf{Z}}\sum_{y\in\mathbf{Y}}\sum_{x\in\mathbf{X}} p(x,y,z)\log_2\left(\frac{p(z)\,p(x,y,z)}{p(x,z)\,p(y,z)}\right) \\ &= \sum_{y\in\mathbf{Y}}\sum_{x\in\mathbf{X}} p(x,y)\log_2\left(\frac{p(x,y)}{p(x)\,p(y)}\right) \\ &= I(X;Y) \end{aligned} \tag{5.13}$$

Similarly we have

$$\begin{aligned} \min(H(X,Z),H(Y,Z)) - H(Z) &= \min((H(X)+H(Z)),(H(Y)+H(Z))) - H(Z) \\ &= \min(H(X),H(Y)) \end{aligned} \tag{5.14}$$

So we see that $NMI(X,Y)$ and $NCMI(X,Y \mid Z)$ can be compared. Now using NMI and $NCMI$ we can test for some types of causal influence.

Assumption 4 real world imperfection.

We have said that we are assuming the presence of deterministic relationships, the tests for causal influence we are going to introduce will not work with perfect deterministic relationships. However, as anyone who has conducted experiments will confirm, experimental data and, by implication, observational data are not perfect. Measurement error is always present to some extent and with observational data it is possible that important variables have not been measured. If we assume that measurement error is random then the values for the variables we have will not be the *actual* value, but rather than value with some measurement error. That is, we want the value of x, but what we get is $x'$where $x' = x+\varepsilon$. ε is the measurement error and we can assume that it varies for each measurement for a given variable and for measurements of different variables in a random manner.

Assumption 5 The dataset is an unbiased, if possibly incomplete, reflection of the underlying causal system.

This is a basic requirement of all learning systems, unless any bias in the dataset is known, it is difficult to compensate for its effects. It has been shown [Spirtes et al 1995], that it may be possible to detect the presence of selection bias, but the LUMIN program does not attempt this.

In essence selection bias is just a variation on the 'Garbage in, Garbage out' adage. If in an analysis to discover the factors that effect fuel consumption in cars, the database contained only information on red sports utility vehicles and green two door cars, then a reasonable discovery would be that colour has a significant impact on fuel economy in cars!

5.3.2 Effects of Causal Influence

So far we have mostly considered causal relationships between two variables. The range of possible causal relationships with two variables is shown in figure 5.4. A simple test like NMI can only determine the difference between independence, case 1, and the existence of some relationship, cases 2, 3 and 4 in figure 5.4. This is because any deterministic relationship between two variables would be expected to cause them to share some mutual information. Since mutual information is a symmetric measure it can tell us nothing about the direction of influence in this simple two variable situation. The situation becomes more interesting when we examine relationships between three variables. Figure 5.5 shows some causal relationships between three variables A, B and C. We can examine the different cases to see what we expect in terms of the mutual information the variables share, and how we would expect that to change when conditioning on B:

- If variable A is a cause of variable B and variable B is a cause of variable C (cases 1 & 3 in figure 5.5) then:
  - A, B, and C all share some mutual information. We would expect the MI of both that between A and B, and B and C, to be greater than or equal to that between A and C[8].
  - If the value of B is fixed and both B and C have more than a single cause[9], and A is only a cause of C through B[10] then A and C will cease to share mutual information. So, we would expect the NCMI of A and C conditioned on B to be significantly less than the NMI of A and C.

This is similar to the CCC causality test, [Cooper 1997, Silverstein et al 2000] used by LCD and other causal discovery systems. This rule allows the potential detection of likely causal chain

[8] This forms part of one test used in the heuristic link removal.

[9] Assumption 4 will suffice as an additional pseudo cause if necessary.

[10] This isn't strictly necessary A can have some causal influence on C in other ways as long as its major influence on C is through B.

relationships, $A \rightarrow B \rightarrow C$ or $A \leftarrow B \leftarrow C$, but on its own it does not allow for a determination of the direction of influence.

- If variable B is a cause of variable A, and variable B is a cause of variable C (case 4 in figure 5.5) then:
  - A, B, and C all share mutual information. We would expect the MI of both that between A and B, and B and C, to be greater than or equal to that between A and C.
  - If the value of B is fixed and both A and C have more than a single cause, and A and C have no common cause other than B, then A and C will no longer share mutual information. So, we would expect the NCMI of A and C conditioned on B to be significantly less than the NMI of A and C.

This again is a subset of the CCC causality test, it tells us that conditioning on B and seeing a significant drop in the mutual information shared between A and C only limits us to one of three possible causal influence configurations, namely $A \rightarrow B \rightarrow C$, $A \leftarrow B \leftarrow C$, or $A \leftarrow B \rightarrow C$. However, this does mean that if we subsequently discover that we have either $A \rightarrow B$ or $C \rightarrow B$ we then know the direction of the other link, that is, $B \rightarrow C$ and $B \rightarrow A$ respectively.

- If variable A is a cause of variable B, and variable C is a cause of variable B (case 2 in figure 5.5) then:
  - A and B, and B and C will share mutual information, but A and C may not.
  - If the value of B is fixed then A and C will share mutual information. So, we would expect the NCMI of A and C conditioned on B to be significantly more than the NMI of A and C.

This is a variation on the CCU test [Silverstein et al 2000]. Once again the LUMIN program checks the MI of A and C conditioned on B and looks for a change in the MI between A and C.

So far in the three variable case we've ignored the possibility of direct causal influence between A and C. Figure 5.6 shows the two possible cases of this type of relationship. Case 1 in figure 5.6 is a simple causal loop. The best we can hope for in this instance is that conditioning on any of the variables significantly reduce the MI between the other two, and all this will tell us is that we have a loop. Similar to cases 1 and 3 in figure 5.5 we will not know the direction of the loop unless we can find the direction of influence of one of the links by other means, then

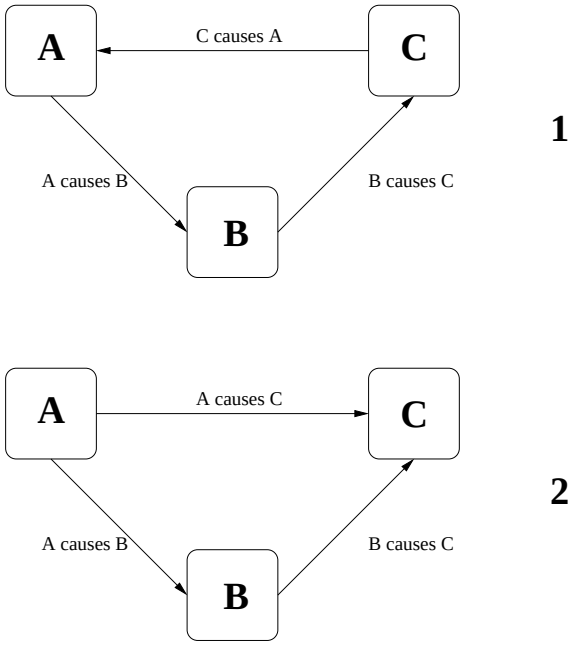


Figure 5.6: Loop Relationships with 3 Variables

we know the direction of all of the links[11]. Case 2 in figure 5.6 is similar to case 2 in figure 5.5 in that the direction of causal influence between A and C, and between B and C may be determined, but the direction of causal influence between A and B will need additional information to determine. The important point here is that while the loops shown in figure 5.6 might make it more difficult to determine the direction of causal influence, they should not introduce incorrect results. However, there are issues that arise with effects that share the same cause see section 5.4.1.

Latent or hidden variables have not been taken into account thus far. An obvious question is what effect might they have on the discovery process? The issue of complexity arises here, with any set of data there could be any number of hidden variables with any number of possible configurations. Hidden variables that do not have a causal influence on the variables being considered can be ignored, a hidden variable which only interacts with a single variable being considered should not introduce any errors into the discovery process. This leaves those cases shown in figure 5.7. The node marked H in each case is assumed to be hidden, that is, while we can see its effects on the other variables we do not have data for it and hence cannot use it to compare MI or as a conditioning variable. We can examine the cases in figure 5.7 one at a time

[11] In practise what we tend to get is a double headed arrow between all parts of a loop.

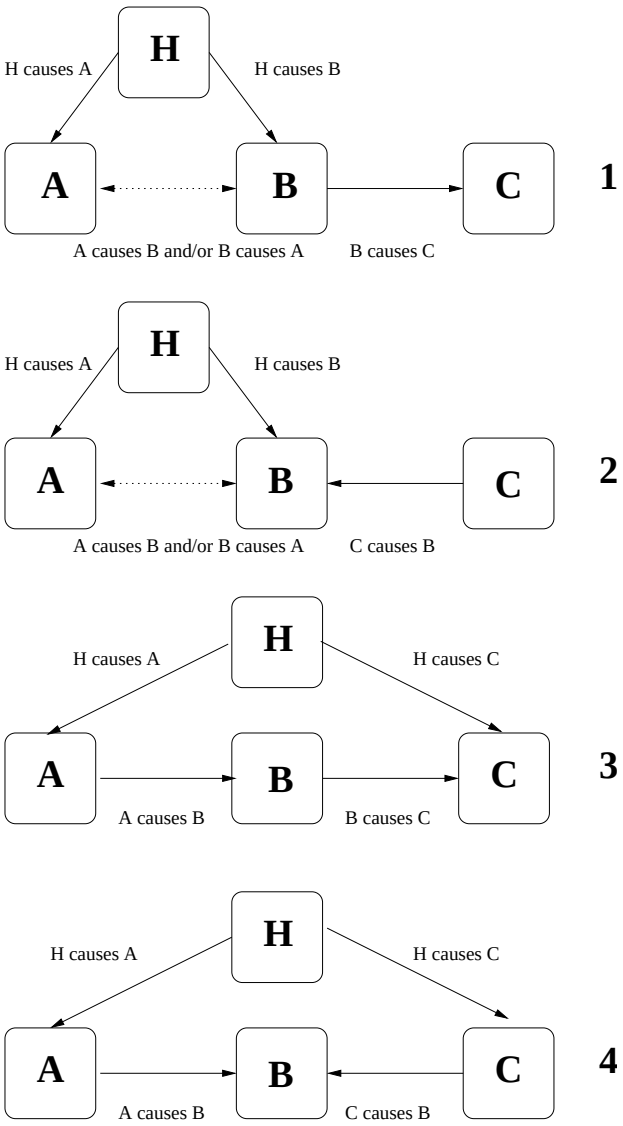


Figure 5.7: Some Hidden Variable Possibilities

Case 1 in figure 5.7 the dashed double headed line indicates that the causal influence between A and B could be either from A to B, from B to A, or there may be none at all. In this case there is still a causal link between A and C, albeit through H. So we would still expect $NCMI(A,C \mid B) < NMI(A,C)$, but perhaps not enough less to help determine causal influence. This situation, without the hidden variable H, could appear to be either of cases 1 and 4 from figure 5.5.

Case 2 in figure 5.7 the dashed double headed line indicates that the causal influence between A and B could be either from A to B, from B to A, or there may be none at all. In this case if the causal influence is actually from B to A, we would expect a drop in MI when conditioning on B, that is we would expect $NCMI(A,C \mid B) < NMI(A,C)$. However, $H \rightarrow B \leftarrow C$ forms a collider, thus if it were possible to perform MI tests with H we would expect that $NCMI(H,C \mid B) > NMI(H,C)$. Now $H \rightarrow A$ so in this case, ignoring any possible direct causal influence between A and B, we might expect $NCMI(A,C \mid B) > NMI(A,C)$, which is also what we would expect if the causal influence was actually $A \rightarrow B$. So, in this instance it's possible we would deduce an incorrect, or at least only partially correct influence from A to B. This should only occur when the causal influence of B on A is significantly weaker than the causal influence of both H on A and H on B. This situation, without the hidden variable H, could appear to be either of cases 2 and 3 from figure 5.5.

Case 3 figure 5.7 shows a situation analogous to cases 1 or 3 from figure 5.5. In this case without the hidden variable H, we would expect there to be a significant drop in MI between A and C when conditioned on B, that is, $NCMI(A,C \mid B) < NMI(A,C)$. However, the presence of H might mask the relationship between A and C if the influence of H on both A and C is greater than that of A on C.

Case 4 figure 5.7 shows a situation analogous to case 2 from figure 5.5. In this instance without the hidden variable H, we would expect that $CNMI(A,C \mid B) > NMI(A,C)$ and we would still expect to see this effect.

So, in the majority of cases the presence of a hidden variable either causes no problem or makes the discovery of the causal influence more difficult. However, in case 2 it could cause the discovery of an incorrect causal relationship. At this point it is unclear how to avoid this situation, the

only comfort is that this is only likely to happen in situations where the causal influence between A and B is weak.

5.3.3 Feature Subset Selection

An intended use of the LUMIN program is for the analysis of observational datasets. It is likely that an observational, as opposed to experimental, dataset will contain attributes that are not relevant to the required analysis. The initial steps of the LUMIN program tests the MI between all the supplied attributes. Those which share a significant amount of MI are linked and it is the search for a direction for these links which forms the basis of the algorithm. This initial MI test acts effectively as an FSS filter which will remove attributes which appear to be unrelated to others in the dataset. A more sophisticated form of FSS could be implemented which, after the initial creation of undirected links, removed all attributes not connected to the attributes of interest. This would not cause any reduction in the effectiveness of the algorithm since new links are not added after this point.

5.3.4 Constructing the Basic Algorithm

Table 5.1 shows the expected changes in the normalised MI values between variables A and C, when the variables are causally linked as shown by the appropriate case from figure 5.5, and the MI is calculated conditioned on B. This is the method used by the LUMIN program to determine possible causal relationships.

Table 5.1 can be used to illustrate the process used by LUMIN. It can be seen that when conditioning on the B variable, the only time we would expect the MI between the A and C variables to increase is when the causal chain represents an instance of that shown in figure 5.5 case 2. So, conditioning on the B variable and observing an increase in the MI between the A and C variables allows the LUMIN program to determine a likely causal relationship between the three variables.

Underlying Situation from figure 5.5	Expected Change between $I(A,C)$ and $I(A,C \mid B)$
Case 1	reduced
Case 2	increased
Case 3	reduced
Case 4	reduced

Table 5.1: Effects of the Causal Assumption

The issue is not as clear cut when the MI between A and C decreases when conditioned on B. To clarify this situation in this case additional information is required.

When there is a causal relationship between the variables A, B and C, when conditioned on B we would expect to see some significant change in the MI between A and C. While a significant decrease in the MI between A and C when conditioned on B cannot determine the underlying relationship it does restrict the possibilities to cases 1, 3 and 4 of figure 5.5. If it is determined later that the direction of either link is towards B, then we know, from cases 1 and 3, that the direction of the other link must be away from B.

The basic algorithm used is shown in algorithm 5.1. In addition to the discovery of causal link direction, LUMIN allows hints to be supplied by the domain and ranking mechanism.

> **Definition 69. Domains** are a grouping mechanism to allow the direction of a causal link, if present, to be predetermined. Each domain has a unique name.

The LUMIN program allows for the creation of zero or more domains. A variable can belong to one or more domains, all variables belong to the 'General' domain with a default ranking of 10. For each domain a variable belongs to, it will have a ranking.

> **Definition 70.** A **ranking** is a positive integer value which applies to a variable within a given domain.

If a link exists between two variables which share a common domain then the direction of the link will be assigned to go from the variable with the lower ranking to the variable with the higher ranking. If they both have the same ranking then the link remains undirected and the program will attempt to assign its direction by the usual mechanism. A pair of linked variables may share multiple domains, but if the rankings in two of the shared domains indicate opposite directions for the link, then the program will terminate with an error. Domains do not affect the creation of links.

The LUMIN program uses a more complicated algorithm than that shown which includes checking domain rankings and forbidden links and adding in specified relationships and dealing with potentially double headed arrows, however, the important details are the same.

So, the LUMIN algorithm uses two measures of statistical significance, one to determine if two variables are related and another to determine if a variable has a significant impact on the strength of the relationship between two other variables. The threshold value of both of these

Algorithm 5.1 The LUMIN Algorithm

Input: A dataset, S, in extended MLC++ format, ie a names file, a CSV data file and an optional domain information file. This describes a set of variables V and their domains D with rankings d_r

Output: A possible causal graph linking a number of the variables in the dataset

1. For all variables $v \in V$ add a node $N_v$to the causal graph
2. For all variables $\{x,y\} \in V : x \neq y$
   If $NMI(x,y) > SigMI$ add an undirected link between N_x and N_y to the causal graph
   (a) If $d \in D, x \in d, y \in d$ then
      i. If $d_r(x) < d_r(y)$ direct the link from N_x to N_y
      ii. If $d_r(y) < d_r(x)$ direct the link from N_y to N_x
3. For all variables $\{x,y,z\} \in V : x \neq y, y \neq z$
   If N_x is linked to N_y and N_y is linked to N_z then
   (a) If $NCMI(x,z \mid y) > MI(x,z)$, we are dealing with figure 5.5 case 2 then
      i. AssignLink(x,y,forwards)
      ii. AssignLink(y,z,backwards)
   (b) If $NCMI(x,z \mid y) < MI(x,z)$, we are dealing with one of cases 1, 3, or 4 from figure 5.5
      i. If either link is currently assigned and pointing towards N_y
      Assign the other link to point away from N_y
      ii. Else Add the pair of links {Link(N_x,N_y),Link(N_y,N_z)} to the LinkList
4. Write out graph of nodes and links

1. Procedure AssignLink(a,b,newDirection)
2. If Link(a,b) is undirected or Link(a,b) has direction newDirection
   (a) Link(a,b) set direction to newDirection
   (b) Search through LinkList
      i. If {Link(a,b),Link(b,c)} is a member of LinkList and newDirection is forwards
         A. Remove {Link(a,b),Link(b,c)} from LinkList
         B. AssignLink(b,c,forwards)
      ii. If {Link(c,a),Link(a,b)} is a member of LinkList and newDirection is backwards
         A. Remove {Link(c,a),Link(a,b)} from LinkList
         B. AssignLink(c,a,backwards)
3. Else remove Link(a,b)

measures can be chosen at runtime by the user. In one sense these values are percentages, their allowed ranges are $[0.0-1.0]$. As with all statistical measures there is no right value, the optimal values can vary with different datasets and this is one of the reasons that LUMIN can show NMI values found.

5.4 Pruning the Graph

Overfitting is a general problem for all inductive learners and there is no reason to believe the LUMIN program is immune from this weakness. The difficulty here is that we do not know of any general analytical method of choosing an optimal value for the magnitude of MI between two attributes to be considered significant. There are many factors which can effect this including: the number of distinct values each attribute can have; the distribution of instances between these values; and the number of joint instances available. The analysis of the selection of an optimal significance measure is beyond the scope of this thesis.

Another problem is that in a deterministic world if *A* causes *B* and *B* causes *C*, then it is absolutely true that *A* is a cause of *C*. This gives rise to the graph shown in figure 5.8. That is

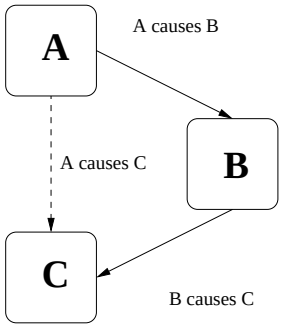


Figure 5.8: Unwanted Indirect Causal Links

we can end up with a direct causal link from *A* to *C* in addition to the indirect link from *A* to *C* through *B*. This problem results from the tests LUMIN uses finding any causal link between variables not just direct causal links. We have developed some heuristic tests to help reduce this type of problem.

5.4.1 Sibling Relationships

Where there are two siblings which share a single parent, as in figure 5.5 case 4, then it can appear that there is a causal relationship between them giving rise to figure 5.9. In this situation it can appear that there is a causal relationship between *C* and *B* whereas their only connection

is through their common parent *A*. LUMIN has a heuristic which allows the removal of such extraneous links when the strength of the link between the parent, in this case *A*, and both the children, in this case *B* and *C*, is stronger than the link between the children.

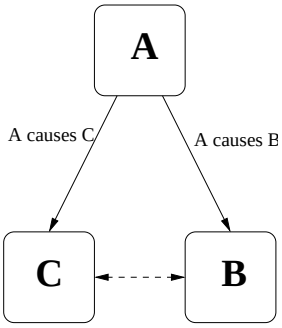


Figure 5.9: Strong Sibling Relationship

5.4.2 Dominant Ancestors

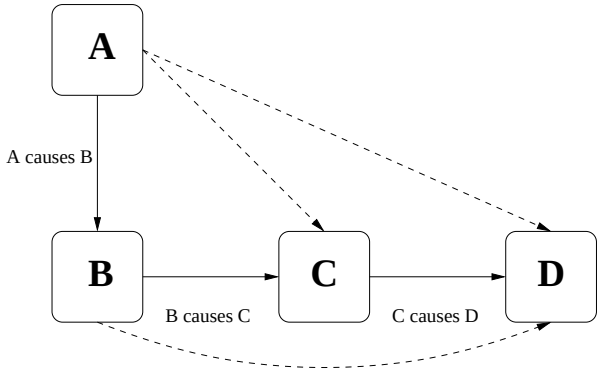


Figure 5.10: Overpowering Ancestor Problem

It is possible that an ancestor is such a strong influence on its decedents that it appears to have some direct causal influence even on its indirect descendants. This commonly occurs when there are no other significant influences on its descendants. Figure 5.10 is an example of such a situation. There is a simple causal chain from *A* to *B* to *C* to *D*, but statistically it appears as though there is a direct relationship between *A* and all of its descendants, and similarly there appears to be a direct relationship between *B* and *D*. LUMIN has a heuristic which allows the removal of dominant ancestor relationships. The heuristic removes ancestor links in the case where the direction of influence to a node is the same from its direct parent and ancestor, and consistent[12] in the chain between the current node and its ancestor, and all the links in the chain between the ancestor and the current node are stronger than the direct link between the ancestor

[12]This includes traversal of bidirectional and undirected links.

and the current node. In figure 5.10 the heuristic would remove the link $A \rightarrow D$ if all the links $A \rightarrow B$, $B \rightarrow C$ and $C \rightarrow D$ were stronger than $A \rightarrow D$. Similarly $A \rightarrow C$ and $B \rightarrow D$ might be removed.

5.5 Computational Complexity

The initial graph construction of the algorithm requires the calculation of the normalised mutual information between all the variables in the database. So, assuming we have N variables and R records the initial computation of NMI between each of the N variables requires going through the dataset once and calculating the NMI between all possible pairs. With N variables each could be paired with $N-1$ other variables, giving a total of $N(N-1)$ pairs. However, this calculation counts each possible pair twice, once each from the perspective of each variable. So, the initial work to discover possible relationships is $O(N(N-1)/2)$ calculations of NMI required. Each NMI calculation requires checking each record once giving a computational complexity of $O(R \cdot N(N-1)/2)$. This is the calculation of NMI is required by the test $NMI(x,y) > SigMI$ carried out in section 2 of algorithm 5.1. However, to calculate the probability distribution for each variable also requires that each record be examined giving an additional $O(R \cdot N)$ computations required, and to simplify the process the current implementation maintains a sorted index for each variable requiring an additional $O(R \cdot N \cdot \log_2 R)$ computations. So, the initial computational complexity for the undirected graph is

$$O\left(\frac{R \cdot N(N-1)}{2}\right) + O(R \cdot N) + O(R \cdot N \cdot \log_2 R) = O\left(\frac{R \cdot N(N+1+2 \cdot \log_2 R)}{2}\right) \quad (5.15)$$

Once the initial NMI calculations have been done there will be a number of links where the NMI value was equal to or greater than the required value. The calculation of the NCMI values will then need to be performed. If we assume that each variable has L links, that is has L relationships to other variables that are considered significant, then if we consider a variable, B, as the centre of a triplet, that is the variable is considered with two of its links to other variables in an $A-B-C$ configuration. If the variable B has L links it will be in the centre of $L(L-1)/2$ triplets. Assuming all N variables have L links there will be a total of $L \cdot N(L-1)/2$ triplets. This calculation of NCMI for each triplet is carried out in section 3 (a) and (b), although we only need to perform the calculation once, of algorithm 5.1. In the worst case of a complete graph this would mean the we would have $N(N-1)(N-2)/2$ triplets, and it would equate to a computational

complexity $O(R \cdot N(N-1)(N-2)/2)$. The worst case of a complete graph should be very rare. Suppose we assume every variable has 3 links[13], then the computational complexity of the NCMI calculations for the graph would be $O(3 \cdot R \cdot N)$. So the total computational complexity for the causal discovery process would vary with

$$O\left(\frac{R \cdot N(N+7+2 \cdot \log_2 R)}{2}\right) \approx O\left(R\left(N^2 + N \cdot \log_2 R\right)\right) \tag{5.16}$$

and in a complete graph we would have total computational complexity that varied with

$$O\left(\frac{R \cdot N\left(N^2 - 2 \cdot N + 3 + 2 \cdot \log_2 R\right)}{2}\right) \approx O\left(R\left(N^3 - N^2 + N \cdot \log_2 R\right)\right) \tag{5.17}$$

This does not take into account the pruning operations, but it should give a general idea of how the computational cost varies with both the number of variables and records.

5.6 Known Limitations

The current design of the LUMIN program has some limitations. The adage 'Garbage In, Garbage Out' applies to the LUMIN program. Like any learner LUMIN depends on the quality of the data it is given. Inaccurate measurements, systematic errors, poorly designed experiments or observations, would result in a dataset that could all lead to the learning of wildly inaccurate causal maps.

5.6.1 Local Discovery Limitations

To some extent all local discovery algorithms suffer from the problem that it is possible that what seems like the best solution when only considering part of the data, is obviously less than optimal when considering all of it. This is the unavoidable trade off in reducing the complexity from considering all of the graph at once to considering only parts of the graph in turn. In addition there are other possible problems mentioned below.

5.6.2 Hidden Variables not Considered

The current version of the LUMIN program does not directly consider the possibility of hidden variables, variables which form part of the system under investigation, but which are not recorded

[13] Any assumption can be considered unreasonable, but in practise assuming every variable has 3 relationships would match or exceed the complexity of much of the real world data of which we are aware.

in the data. When two variables appear to exert a causal influence on each other a double headed arrow is used. However, a possibility which could be explored is that both variables have a common hidden cause, H, as shown in figure 5.11. This would cause additional complexity as

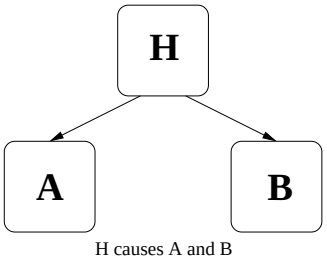


Figure 5.11: Hidden Common Cause

introducing possible hidden variable discovery requires that a number of possible causal graphs be explored. Then it would be necessary to find some method of choosing how many and which graphs to present.

5.6.3 Limitations of the Statistical Tests

It is obvious that statistical tests provide information on statistical relationships. In general the accuracy of statistical tests improve with increasing amounts of data. Thus we would expect the performance of the LUMIN program to improve with increasing amounts of data. Statistical relationships can occur where no causal relationship exists. Thus regardless of how much data is available or how strong a statistical relationship appears to be, there is no guarantee it represents a causal relationship. In general, however, a causal relationship will give rise to a statistical relationship.

5.6.4 No Explicit Representation of Time

The LUMIN program has no explicit method of representing time sequence data. Time separation between different variables can be represented through domain rankings, but time separation between instances of the same variable can only be achieved by folding the dataset, see section 2.15.

5.7 Implementation Issues

The current implementation of the LUMIN algorithm is not perfect. Our aim was to have a learner which we could use to learn form 'standard' datasets so we could gauge its basic abilities.

This lead to design choices which would need to be revisited if the learner were to be used more generally.

5.7.1 Limited Data Types

The LUMIN program only knows about three type of data, integers, real numbers and strings. While it is reasonably flexible in allowing numerical limits to be set, specifying allowed values, and defining whether the values are to be treated as nominal or ordinal[14], more data types and a more flexible treatment of strings might be useful in some circumstances.

5.7.2 Memory Requirements

The current implementation keeps the entire dataset in memory. This is not suitable for very large datasets and it would be advantageous to have some method of using alternative storage to permit the analysis of very large datasets. The program is fairly efficient with its memory usage so that the maximum required memory when processing the ALARM dataset was under 11MB.

5.7.3 Database Access

The current LUMIN program only accepts its data in a flat file format. Accessing data from databases in a standard table format using a common access method like ODBC could prove useful.

5.7.4 Computational Efficiency

The current implementation it is not particularly efficient. More work in this area would allow for better performance on large datasets. Its worth noting that, possibly due to performing a lot of comparative tests, LUMIN runs more quickly if strings in the data are represented by integers. Even so the performance is acceptable, when run on a lightly loaded AMD Athlon(tm) 64 X2 Dual Core Processor 4400+ processor, using the 20000 record ALARM dataset, the program completed in under 8 minutes, it took a similar time for the alternate 10000 record ALARM dataset. For comparison the Grow-Shrink learner from the Bnlearn R package under similar conditions took about 10 seconds and just under 8 minutes for the 20000 and 10000 record data sets respectively.

[14]Only alphabetic ordering is currently supported for strings.

5.8 Summary

This chapter has outlined our goals when we set out to design a new causal learner. The view, based on our understanding of causality, that causally related variables will share mutual information, allows the simple and direct use of that measure in determining the presence and direction of causal relationships. This leads to the core of the algorithm being computationally fairly simple. We believe LUMIN represents a unique alternative causal learner, without the restrictions imposed on most learners by the causal Markov condition. LUMIN can potentially discover relationships of any type including feedback loops, and should be able to do so tractably even on large datasets. In addition to basic causal discovery LUMIN can use a wide variety of domain knowledge. Some learners allow the inclusion of domain knowledge, such as forbidden arcs and specific directions for some arcs. LUMIN will allow the inclusion of any level of link specific domain knowledge, and its domain and ranking capabilities allow for considerable flexibility. LUMIN's learning the direction of causal influence takes place after the supplied domain knowledge has been taken into account, allowing it to effect the learning process without needing any additional computation.

Chapter 6

A Comparison of Learners and what is Learnt - Known Structure

---

This chapter examines what is learnt by a number of different learners when applied to datasets where the underlying causal structure is known. In the *real world* very little is known with absolute certainty. So, to test learners against known relationships, specific models are usually constructed and data generated from these models. This allows perfect knowledge of what the learner is trying to reconstruct and thus allows for comparison between learners of the accuracy with which they recreate the original model. The first part of this chapter introduces the Bnlearn R package and explains our reasons for using it. We then introduce a novel metric to measure the similarity of two causal networks. The main part of this chapter examines a number of datasets, three produced by 'well known' causal networks: ALARM, Hailfinder and Insurance, and others constructed specifically to test non-linear relationships and feedback loop discovery, which are part of the design goals for the LUMIN learner. Each of these networks is examined in turn. Basic causal discovery is tested using the three 'well known' networks for each of which two datasets are analysed by three *standard* causal learners and the LUMIN learner. Each of the datasets for these networks comes from a different source to minimise any bias or unintended property of any single dataset. The output graph(s) of each learner are scored against the correct graph so that the accuracy of the learners can be compared. The same learners are all used on the non-linear, and feedback loop networks. The tests with these networks use only a single dataset as we had no independent means of generating the data. Since a large number of graphs were

produced, for clarity, only some are shown in this chapter and the remainder appear in Appendix B.

In this chapter we do not consider dynamic BNs, DBNs, as although there has been some research into learning DBNs from data [Ghahramani 1998, Peña et al 2005] they tend to require a significant amount of domain knowledge, such as node ordering, the data to be presented in time order, or an outline of the network structure. We know of no general DBN learner that operates with no additional domain information and so we do not feel that their inclusion would add any value to the current analysis, focusing on basic causal discovery, as not all the causal learners can make use of the same domain information, making a comparison between them difficult. Even in the case of the test networks which include feedback loops, where the DBNs should be better able to cope than the standard learners, the majority of causal learners would be at a disadvantage without domain knowledge, making it difficult to determine how much of the difference was the ability to cope with a loop and how much was due to the increased domain information. Since LUMIN should be able to discover feedback loops even when supplied with no domain information it allows a direct comparison with other general causal learners even when feedback loops are involved.

6.1 The Bnlearn R Package

Bnlearn[1] is a 'well known' library package for the statistical computing language and environment, R [2]. Bnlearn written by Marco Scutari[3] is an R package for learning the graphical structure of Bayesian networks. The package implements a range of learners including constraint-based, search-and-score and hybrid learners. Each learner has a range of appropriate arguments and options, and the learner's implementations allow them to work with, at least, moderately complicated networks. In addition to the learning algorithms the library also includes representations of a number of 'well known' networks with the ability to generate datasets for these networks. It is possible to export these generated datasets in a simple format, and additional datasets can be imported using the same simple format. In addition Bnlearn is able to generate images of the graphs learnt by any of its learners using the same software, graphviz, used by LUMIN, and to directly produce images of these graphs in a standard format. The R environment and the Bn-

[1] The software and documentation is available for download from `http://www.bnlearn.com/`.
[2] See `http://www.r-project.org/` for downloads and more details on R.
[3] Marco Scutari can be contacted at `marco.scutari@stat.unipd.it`.

learn package are both freely[4] available and will work under various different operating systems. These factors make it an obvious choice for use in comparative testing for this thesis.

In the tests we used three learners from the Bnlearn package: Grow-Shrink, GS, a constraint-based learner that uses Markov Blankets to discover local structure [Margaritis 2003]; Hill-Climbing, HC, a search-and-score network learner that uses the Bayesian Information Criterion, BIC, for its scoring heuristic; and lastly Max-Min Hill Climbing, MMHC, a hybrid learner using both constraint-based and search-and-score techniques[5].

6.2 Causal Likeness - An Alternative Metric

Comparing the output of causal network learners is obviously a matter of identifying the differences, if any, between the learnt network and the actual network. There is no single universally accepted 'best' method of ranking different networks in terms of their *causal likeness* to the original. Various metrics exist, but for the purpose of this thesis we would like to introduce, what we believe to be, a novel metric which we call the CL metric. Since the intention behind LUMIN is learning causal relationships an appropriate metric is one which rewards the correct discovery of causal relationships and penalises incorrect causal assertions. So, going roughly from good to bad we could classify discovered causal links as follows: correct relationship, both the presence of a link and its direction; correct if limited discovery, there is a link present, but the direction is not known; relationship discovery, a causal relationship has been found, but the direction of causal influence is wrong; failure to discover a causal link; asserting the existence of a nonexistent causal link; asserting the existence and direction of causality of a nonexistent causal link. Table 6.1 gives the relative scores of each possible situation of required relations vs learnt one.

Actual Relationship	Learnt Relationship	Score
$A \rightarrow B$	$A \rightarrow B$	5
	$A - B$ or $A \leftrightarrow B$	4
	$A \leftarrow B$	3
	$A \quad B$	0
$A \quad B$	$A \rightarrow B$	0
	$A - B$ or $A \leftrightarrow B$	1
	$A \leftarrow B$	0
	$A \quad B$	4

Table 6.1: CL Metric for Scoring Learnt Networks

[4]The R environment including source code is freely available under the GNU General Public License, and the Bnlearn package is freely available under Creative Commons Attribution-Share Alike License.

[5]See section 2.9.3 for more information on both the HC and MMHC learners.

The question should be asked about why we don't use a more common measure like the Hamming distance, [Hamming 1950], between the learnt and original graph. The answer is that in causal terms we believe some changes are more significant than others. In Hamming terms $A - B$ is equidistant from both $A \quad B$ and $A \rightarrow B$, but causally we feel it is worse to indicate a causal relationship where none exists, than is it to indicate a causal relationship, without indicating the direction, where one exists. Similarly in Hamming terms $A \rightarrow B$ is equidistant from $A \quad B$ and $A \leftarrow B$. We feel that in terms of causal discovery although $A \leftarrow B$ indicates a causal link in the wrong direction it is far more informative than $A \quad B$ as at least it indicates there is a possible causal relationship which could be worth further study. The motivation behind creating the LUMIN learner was to provide a tool to help with understanding the causal relationships in systems where experiments were not necessarily possible. To this end we feel that the most important aspect is the discovery of the presence or absence a causal relationship, with the direction of causality being secondary. We will show the Hamming distance for comparison for some of the graphs[6].

6.3 Alarm Network

Medical diagnosis is an area in which a significant amount of machine learning research is focused. Medical diagnosis requires the combination of observational data with domain knowledge. These tasks are common and hence providing support for them will provide a useful tool. The ALARM network is a simulation of a patient monitoring system for patients in need of intensive care.

ALARM stands for 'A Logical Alarm Reduction Mechanism'. This is a fairly complex network with 37 attributes linked by 46 relationships. The details of the model are described in [Beinlich et al 1989]. The attributes can be divided into three groups, 8 diagnoses, 16 findings and 13 intermediate variables. Since the Alarm Network is a simulation its details are known. A representation of the network is shown in figure 6.1, the attributes are represented by nodes on the network and their relationships are represented by edges, or links, joining the nodes. The variables in the ALARM network are shown in table 6.2[7].

[6] In practise while the values of Hamming distance and the CL metric differ it was rare for the relative ordering to be different.

[7] The abbreviations and ordering given here are the same as those in the Bnlearn R package and vary from those in the Norsys data.

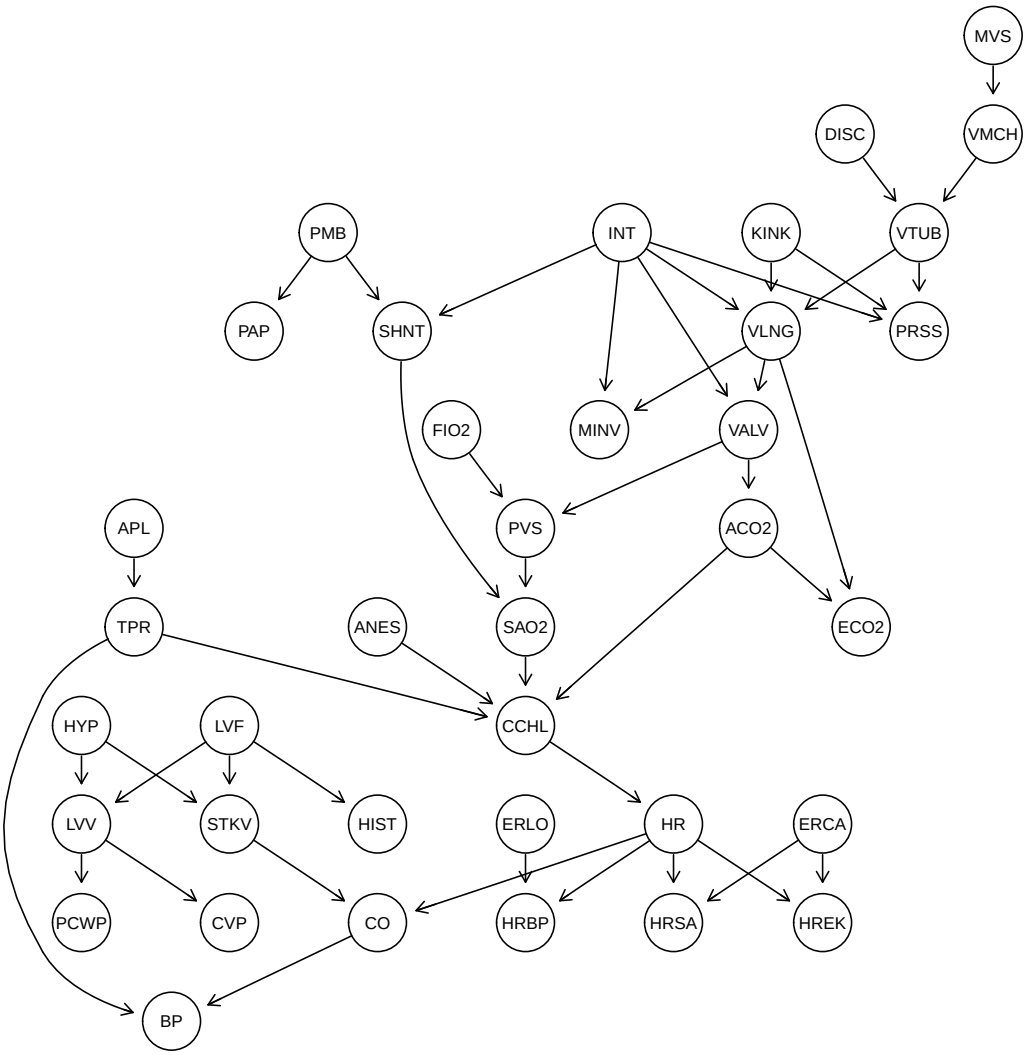


Figure 6.1: The ALARM Network

Table 6.2: ALARM Network Variables

Variable Name	Description	Possible Values
CVP	central venous pressure	LOW, NORMAL and HIGH
PCWP	pulmonary capillary wedge pressure	LOW, NORMAL and HIGH
HIST	history	TRUE and FALSE
TPR	total peripheral resistance	LOW, NORMAL and HIGH
BP	blood pressure	LOW, NORMAL and HIGH
CO	cardiac output	LOW, NORMAL and HIGH
HRBP	heart rate / blood pressure	LOW, NORMAL and HIGH
HREK	heart rate measured by an EKG monitor	LOW, NORMAL and HIGH
HRSA	heart rate / oxygen saturation	LOW, NORMAL and HIGH
PAP	pulmonary artery pressure	LOW, NORMAL and HIGH
SAO2	arterial oxygen saturation	LOW, NORMAL and HIGH
FIO2	fraction of inspired oxygen	LOW and NORMAL
PRSS	breathing pressure	ZERO, LOW, NORMAL and HIGH
ECO2	expelled CO2	ZERO, LOW, NORMAL and HIGH
MINV	minimum volume	ZERO, LOW, NORMAL and HIGH
MVS	minimum volume set	LOW, NORMAL and HIGH
HYP	hypovolemia	TRUE and FALSE
LVF	left ventricular failure	TRUE and FALSE
APL	anaphylaxis	TRUE and FALSE
ANES	insufficient anesthesia/analgesia	TRUE and FALSE
PMB	pulmonary embolus	TRUE and FALSE
INT	intubation	NORMAL, ESOPHAGEAL and ONESIDED
KINK	kinked tube	TRUE and FALSE
DISC	disconnection	TRUE and FALSE
LVV	left ventricular end-diastolic volume	LOW, NORMAL and HIGH
STKV	stroke volume	LOW, NORMAL and HIGH
CCHL	catecholamine	NORMAL and HIGH
ERLO	error low output	TRUE and FALSE
HR	heart rate	LOW, NORMAL and HIGH
ERCA	electrocauter	TRUE and FALSE
SHNT	shunt	NORMAL and HIGH
PVS	pulmonary venous oxygen saturation	LOW, NORMAL and HIGH
ACO2	arterial CO2	LOW, NORMAL and HIGH
VALV	pulmonary alveoli ventilation	ZERO, LOW, NORMAL and HIGH
VLNG	lung ventilation	ZERO, LOW, NORMAL and HIGH
VTUB	ventilation tube	ZERO, LOW, NORMAL and HIGH
VMCH	ventilation machine	ZERO, LOW, NORMAL and HIGH

6.3.1 Results

Outlined below are the results of learning from the test data for the ALARM network. The same data was analysed using a number of machine learners from Bnlearn and also using LUMIN. The relationships discovered in the data are described below. We examine areas of the graphs which caused problems and for the LUMIN learner discuss how altering its NMI and NCMI thresholds varies what is learnt. After going through both datasets we score the networks using both our own novel scoring system and the more standard Hamming distance[8].

6.3.2 Bnlearn R Package - Bnlearn Data

This series of experiments used the 20000 record ALARM dataset generated by the Bnlearn package. The network obtained, using the optimised Grow-Shrink Markov Blanket learner with a target nominal type I error rate, or alpha value, α, of 0.05 is shown in figure 6.2. The network has all of the nodes connected, so it retains 37 nodes, and has 38 relationships between them. The learner notes that a head to head node structure, or v-structure, $CCHL \rightarrow PVS \leftarrow SAO2$ is found, but that one or both of the links have already been directed in the opposite direction. If the target nominal type I error rate is raised to 0.1 it produces the graph show in figure B.1. At the alpha level of 0.1 the following v-structures give rise to errors over the direction of the edges involved: $CCHL \rightarrow HREK \leftarrow ERCA$, $CCHL \rightarrow HRSA \leftarrow ERCA$, $HREK \rightarrow CO \leftarrow STKV$, $HRSA \rightarrow CO \leftarrow STKV$, $CCHL \rightarrow HREK \leftarrow CO$ and $CCHL \rightarrow HRSA \leftarrow CO$. Similarly figure 6.3 is the network learnt by the Hill-Climbing algorithm this graph again retains the 37 nodes, although *ANES* has no relationships, and has 53 relationships. Figures 6.4 and B.2 are the networks learnt by the Max-Min Hill-Climbing algorithm with alpha values of 0.05 and 0.1 respectively. These networks have 44 and 45 relationships respectively.

6.3.3 The LUMIN Learner - Bnlearn Data

This series of experiments used the 20000 record ALARM dataset generated by the Bnlearn package. The LUMIN learner has two tunable parameters, and a number of optional functions. The tunable parameters are the significance level for the normalised mutual information, NMI, that is, the level of mutual information at which two variables are considered related and the relative change in the normalised mutual information between two variables when conditioned on a third variable, NCMI, that is considered significant. These values are both expressed in the

[8] Actually we use a minor variation since with LUMIN the edges $-$, and $\leftrightarrow$ are equivalent.

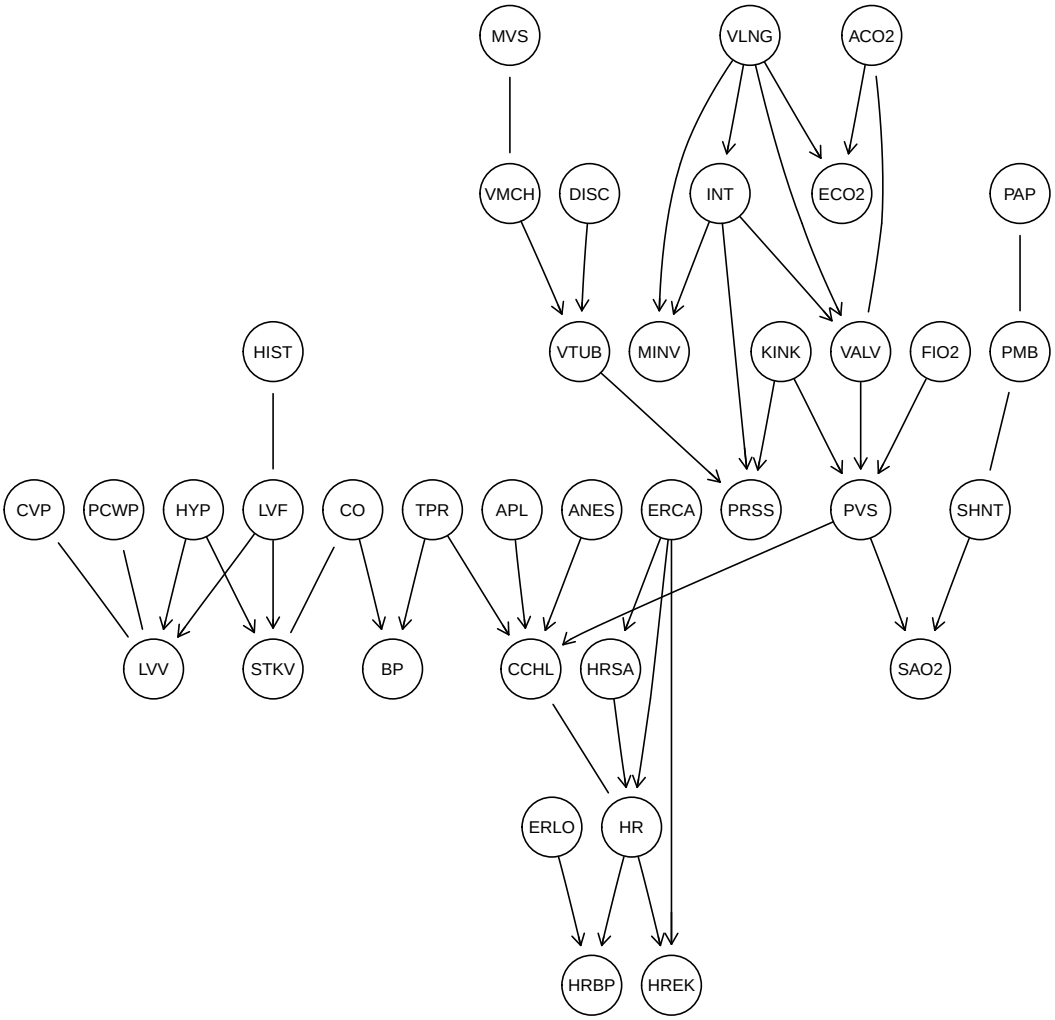


Figure 6.2: ALARM Network (Bnlearn Data) Learnt by Grow-Shrink Learner - $\alpha = 0.05$

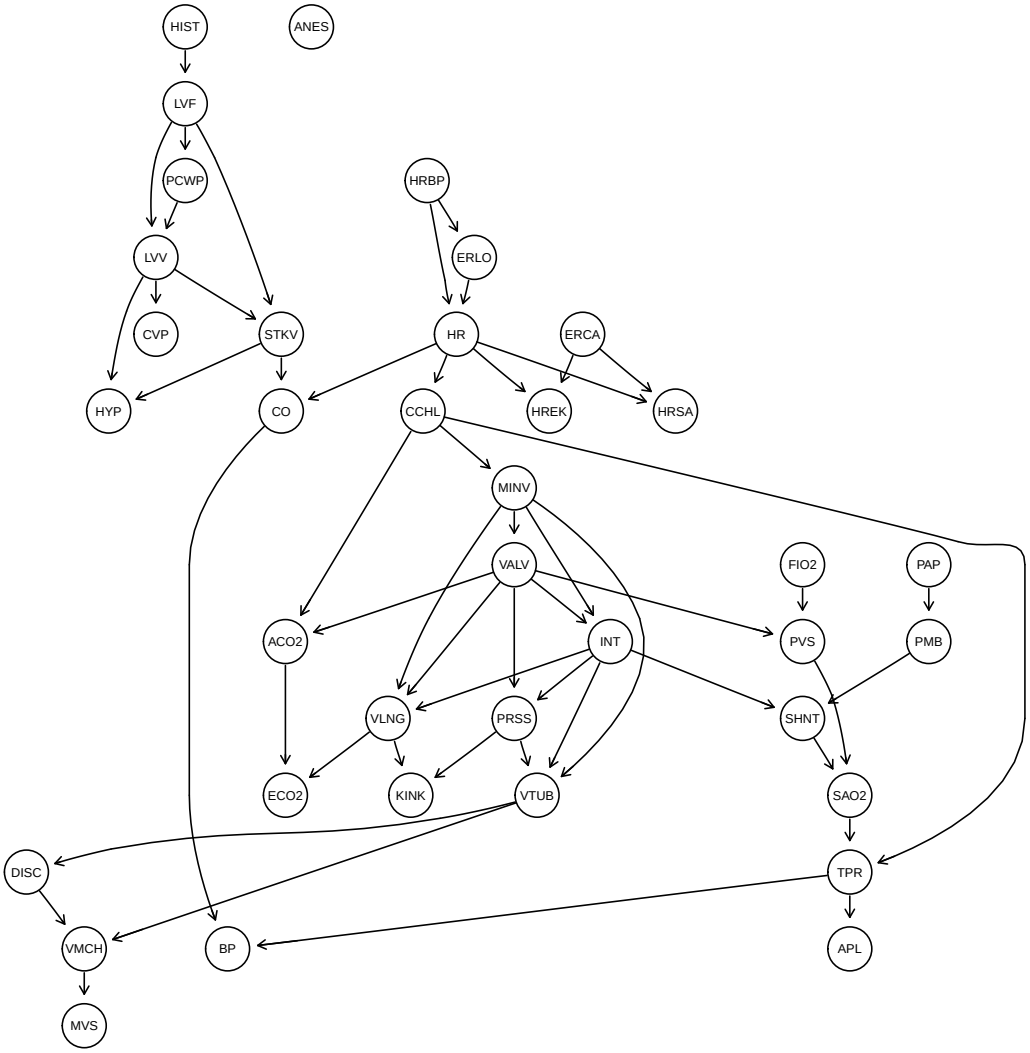


Figure 6.3: ALARM Network (Bnlearn Data) Learnt by Hill-Climbing Learner

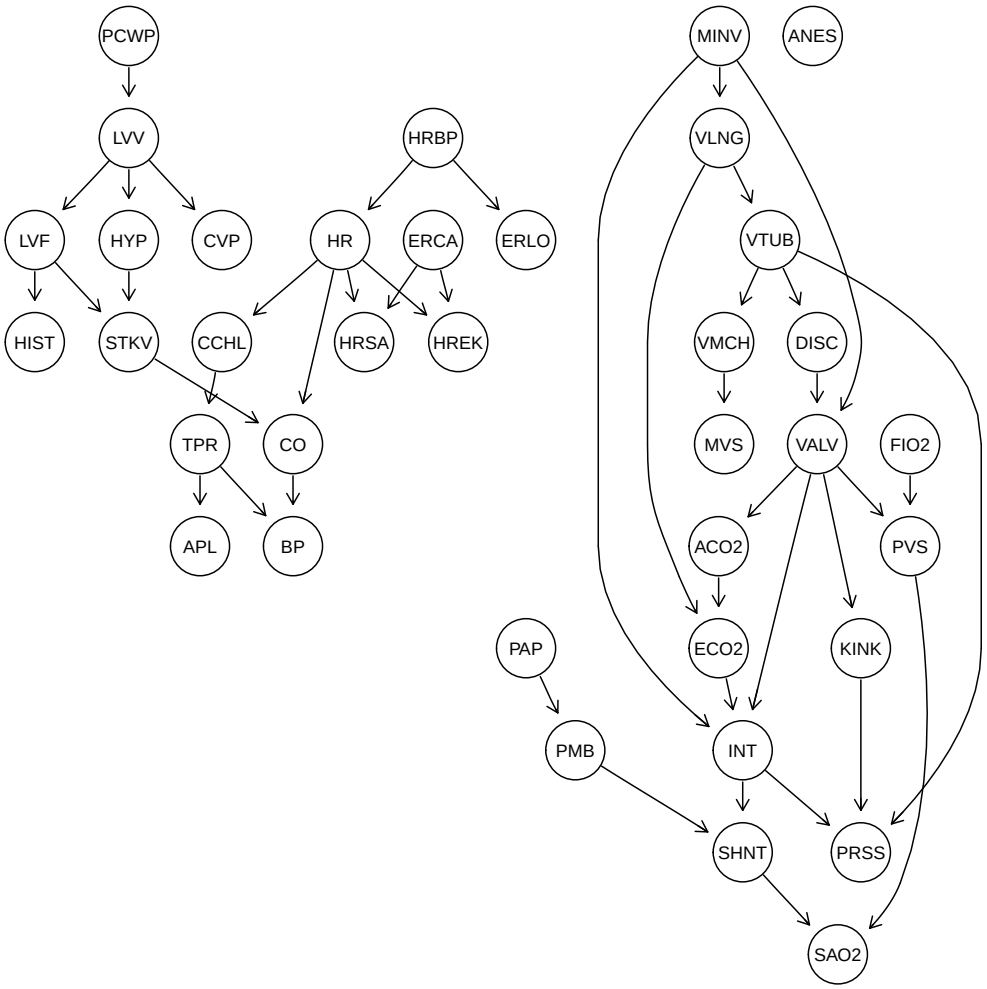


Figure 6.4: ALARM Network (Bnlearn Data) Learnt by Max-Min Hill-Climbing Learner - $\alpha = 0.05$

$[0-1]$ range. Heuristic functions deal with eliminating false relationship discovery in situations where the influence of an ancestor, or parent, is overly strong. There are also options to display or hide unconnected nodes and to remove, most, undirected links.

Figure 6.5 shows the graph of the ALARM dataset produced by LUMIN with a normalised mutual information threshold of 0.2 and a conditional mutual information change threshold of 0.5 without any heuristic clean up. The low significance values and no heuristic pruning leads to many low likelihood relationships. There are 69 relationships shown on the graph. The program discovered contradictory link direction issues with the following twenty one triplets:

HIST–PCWP–HYP	HYP–PCWP–LVF	CO–HRBP–ERLO
HREK–HRBP–ERLO	HRSA–HRBP–ERLO	CCHL–HRBP–ERLO
CO–HREK–ERCA	CCHL–HREK–ERCA	CO–HRSA–ERCA
CCHL–HRSA–ERCA	ECO2–MINV–INT	INT–MINV–PVS
INT–MINV–VLNG	INT–MINV–VTUB	HIST–LVV–HYP
HYP–LVV–LVF	EC02–VALV–INT	INT–VALV–VLNG
INT–VALV–VTUB	MVS–VTUB–DISC	DISC–VTUB–VMCH

Even at this relatively low significance level we can see that the graph has become separated into two parts. This is because despite there being a causal link between *SAO*2 and *CCHL* they share very little mutual information only about 0.005 in normalised MI form. Similarly *CCHL* and *ACO*2 only share about 0.004 of normalised MI, and *ANES* and *CCHL* only share about $9.4x10^{-5}$ in normalised MI. However, TPR and *CCHL* share about 0.52 of NMI and *HR* and *CCHL* share about 0.59 NMI. This demonstrates one issue with this from of simple local constraint-based learners, namely, that even when a relationship exists, if it is only a weak relationship it can be difficult to discover it. *CCHL* has 4 causes, but in terms of NMI 3 are quite weak and so its only the relationships between *CCHL* and both TPR and *HR* that are explored.

Figure 6.5 also shows the effect of a dominant ancestor. In the correct graph shown in figure 6.1 we see that the HYP variable only has a direct influence on the LVV and $STKV$ variables, but in figure 6.5 it appears to be directly related to $PCWP$ and CVP in addition. These variables are only connected to HYP through LVV, but they still share a significant amount of NMI with HYP, about 0.32 in the case of CVP and about 0.50 for $PCWP$. Keeping the same parameters as figure 6.5, but allowing the heuristic clean up, produces the graph shown in figure 6.6. This figure shows additional functionality of the LUMIN program. In figure 6.5 unconnected nodes were

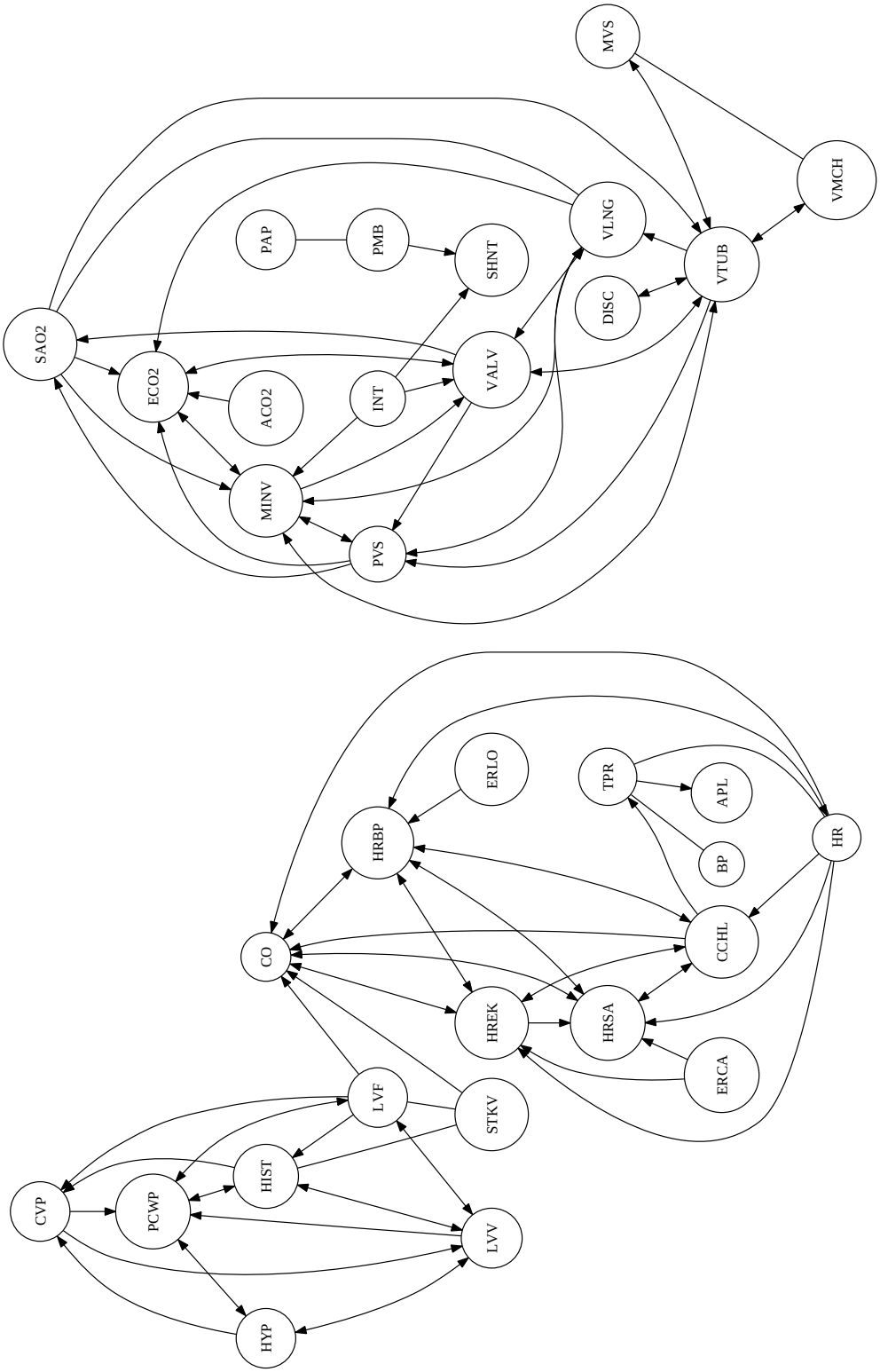


Figure 6.5: ALARM Network Learnt by LUMIN with NMI 0.2 and NCMI $\delta = 0.5$

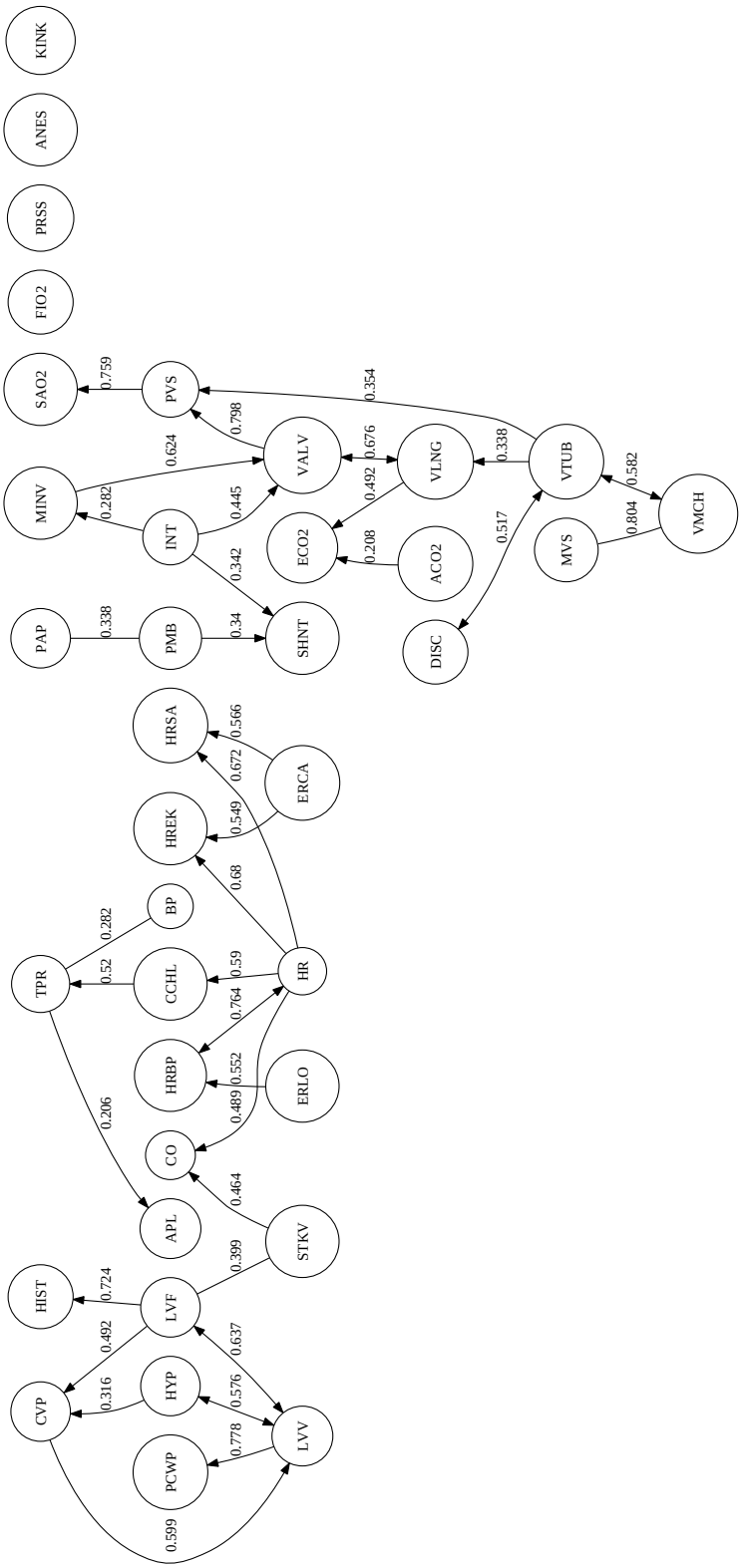


Figure 6.6: ALARM Network (Bnlearn Data) Learnt by LUMIN with NMI 0.2 and NCMI $\delta = 0.5$ with Heuristic Link Removal

not shown on the graph, they are shown in figure 6.6 which indicates that $FIO2, PRSS, ANES$, and $KINK$ are all unconnected to the other nodes in the graph. In addition figure 6.6 shows the NMI value for each relationship next to the link that represents it. Thus although LUMIN could not determine the directionality of the relationship between MVS and $VMCH$ it can be seen as a strong relationship, $NMI = 0.804$, whereas the link between TPR and APL while directed, incorrectly as it happens, shows the relationship to be fairly weak $NMI = 0.206$. Looking at missing links we can use the triplet $HYP \rightarrow STKV \leftarrow LVF$ as an example. The link $HYP \rightarrow STKV$ is missing as the NMI between these two variables is about 0.16, the $LVF \rightarrow STKV$ is present, although without a determined direction, with an NMI of about 0.40.

Heuristic link removal helps in dealing with some of the inaccuracies in the learnt graph, but there are still errors. This is partly due to the low threshold used for determining link direction. Figure B.3 shows the network learnt by the LUMIN program using an NMI threshold of 0.2 for relationship discovery and a NCMI threshold of 0.7 for direction of causal influence discovery. The higher threshold means that fewer links are directed, but hopefully fewer errors in direction occur.

Examining the difference between figures 6.6 and B.3 should be useful in clarifying the effects of raising the threshold for direction determination using the NCMI test. Firstly since the test for the existence of relationships, the NMI threshold, was not changed, the initial graph LUMIN generates will be identical in both cases. So, in the final graph the nodes without any relationships will be identical. In this case the nodes $FIO2$, $PRSS$, $ANES$, and $KINK$ are unconnected in both graphs. Similarly not all nodes relationships have changed between the two graphs. The following nodes have identical relationships in both graphs: PAP, $ECO2$, $PCWP$, PMB, $SHNT$, CO, $STKV$, $ERLO$, PVS, $SAO2$, $DISC$, $ACO2$, $VMCH$, MVS, BP, $ERCA$, $HRSA$, $HREK$, and $HRBP$. Although both graphs agree on the relationships of these nodes, it does not necessarily indicate they they are correct or complete. In both graphs the PAP node has a single undirected link to the PMB node. In the original graph figure 6.1 we see that this link should be directed as $PMB \rightarrow PAP$, the current LUMIN program has no way to determine the direction of this link, the undirected link is the best it can be expected to learn.

We now examine and explain the differences in these two graphs. In figure 6.6 we have $LVF \rightarrow CVP \rightarrow LVV$ in figure B.3 this changes to the correct relationship $LVV \rightarrow CVP$. Similarly in figure 6.6 we have $LVV \leftrightarrow HYP \rightarrow CVP$ and $LVF \rightarrow HIST$ these change to $HYP \rightarrow LVV$ and

$LVF - HIST$ in figure B.3, the correct relationships are $LVV \leftarrow HYP \rightarrow STKV$ and $LVF \rightarrow HIST$. According to our scoring metric in table 6.1 considering only the nodes LVF, CVP, LVV, HYP and $HIST$ and their interrelationships we have gone from a score of 32 to a score of 43. The score for the correct relationships between these nodes would be 44. The points were lost at the NCMI $\delta = 0.7$ level of significance because LUMIN could not determine the direction of the link the link $LVF \rightarrow HIST$. In figure 6.6 the $HIST$ node has relationships with $PCWP$, LVV, $STKV$ and LVF, this reduces to the single relationship $HIST - LVF$ in figure B.3. Figure 6.1 shows that the correct relationship is $LVF \rightarrow HIST$. Due to limitations in this form of local learning the best we can achieve is the directionless $LVF - HIST$ relationship. Overall we can say that, in this case, increasing the NCMI threshold from 0.5 to 0.7 improved the accuracy of the LUMIN learner.

Another set of nodes whose relationships vary between figures 6.6 and B.3 are HR, $CCHL$, TPR, and APL. Considering these nodes and our CL scoring metric in table 6.1 we see that in figure 6.6 they have a score of 21 and in figure B.3 a score of 24. The correct score for these nodes and their relationships is 27. While increasing the NCMI threshold improved the accuracy there are still problems. It would not normally be possible to determine the direction of the link between APL and TPR so the loss of that point is unavoidable. The direction of the link between TPR and $CCHL$ should be able to be determined. However, to make such a determination requires the presence of a link from any of $ANES$, $SAO2$, or $ACO2$ to $CCHL$. These links are missing due to their low NMI values of 9.4×10^{-5}, 0.005 and 0.004 respectively. In addition had any of these missing links been present, and correctly directed, then it would have been possible to determine the direction of the link between $CCHL$ and HR.

Lowering the NMI threshold is not all positive as it can introduce additional extraneous links which can confound the correct discovery of direction in stronger links. Figure B.4 shows the network learnt with an NMI threshold of 0.005 and an NCMI threshold of 0.9. As noted before $ANES$ shares so little mutual information with $CCHL$ that even at this low significance level it has no worthwhile relationships. The low NMI threshold allows the discovery of the relationship between $SAO2$ and $CCHL$. Allowing such weak relationships can cause errors in determining the direction of causality even in stronger relationships. An example of this is the relationship between TPR and APL and between TPR and BP. The correct direction for these links is $TPR \rightarrow BP$ and $TPR \leftarrow APL$. The learnt links are $TPR \leftrightarrow BP$ and $APL \rightarrow TPR$. The direction

of these relationships is confused by the existence of low probability links. Similarly the link $VTUB \rightarrow VLNG$ is correct in figure B.3 but becomes $VTUB \leftrightarrow VLNG$ is figure B.4. A better compromise might be using an NMI threshold of 0.1 and an NCMI threshold of 0.9, the result of this combination is shown in figure 6.7

6.3.4 Bnlearn R Package - Jie Cheng Data

This series of experiments used the 10000 record ALARM dataset generated by Jie Cheng[9]. Figure 6.8 shows the graph learnt by the Grow-Shrink Markov Blanket learner with a target nominal type I error rate of 0.05. There are no isolated nodes in the network which has a total of 65 links. Figure B.22 shows the network learnt by the Grow-Shrink Markov Blanket learner with a target nominal type I error rate of 0.1, this graph also has no isolated nodes and has 71 links. Figure B.6 shows the graph produced by the Hill Climbing learner, there are no isolated nodes and 96 links. Figure 6.9 shows the graph learnt by the Max-Min Hill Climbing learner with a target nominal type I error rate of 0.05 , there are no isolated nodes and 66 links. Figure B.7 shows the graph learnt by the Max-Min Hill Climbing learner with a target nominal type I error rate of 0.1, there are no isolated nodes and 70 links.

6.3.5 LUMIN Learner - Jie Cheng Data

This series of experiments used the 10000 record ALARM dataset generated by Jie Cheng. As with the Bnlearn R package results above this section will just show the results of learning from the alternate ALARM data without an in depth discussion of the results. Figure B.8 shows the graphs learnt by the LUMIN learner with a threshold of 0.2 for NMI and 0.5 for NCMI with no heuristic link removal. In the graphs the nodes *APL*, *ERLO*, and *ERCA* are isolated having no links to any other node. The nodes *TPR* and *BP* are only connected to each other so no determination of the direction of the relationship is possible. In total there are 121 links. Figure 6.10 shows the graph learnt by the LUMIN learner using the same threshold values, but allowing heuristic link removal. In total there are 37 links. Figure B.9 shows the graph learnt by the LUMIN learner with an NMI threshold of 0.2 and an NCMI threshold of 0.7 there are 37 links, for consistency we include figure B.10 with a NMI threshold of 0.05 and an NCMI threshold of 0.9, with such a low NMI threshold there are no isolated nodes, and we have 39 links. Lastly figure 6.11 shows the graph learnt when the NMI threshold is 0.1 and the NCMI threshold is 0.9.

[9]Jie Cheng can be contacted as `jcheng@cs.ualberta.ca` or `j_cheng88@yahoo.com`.

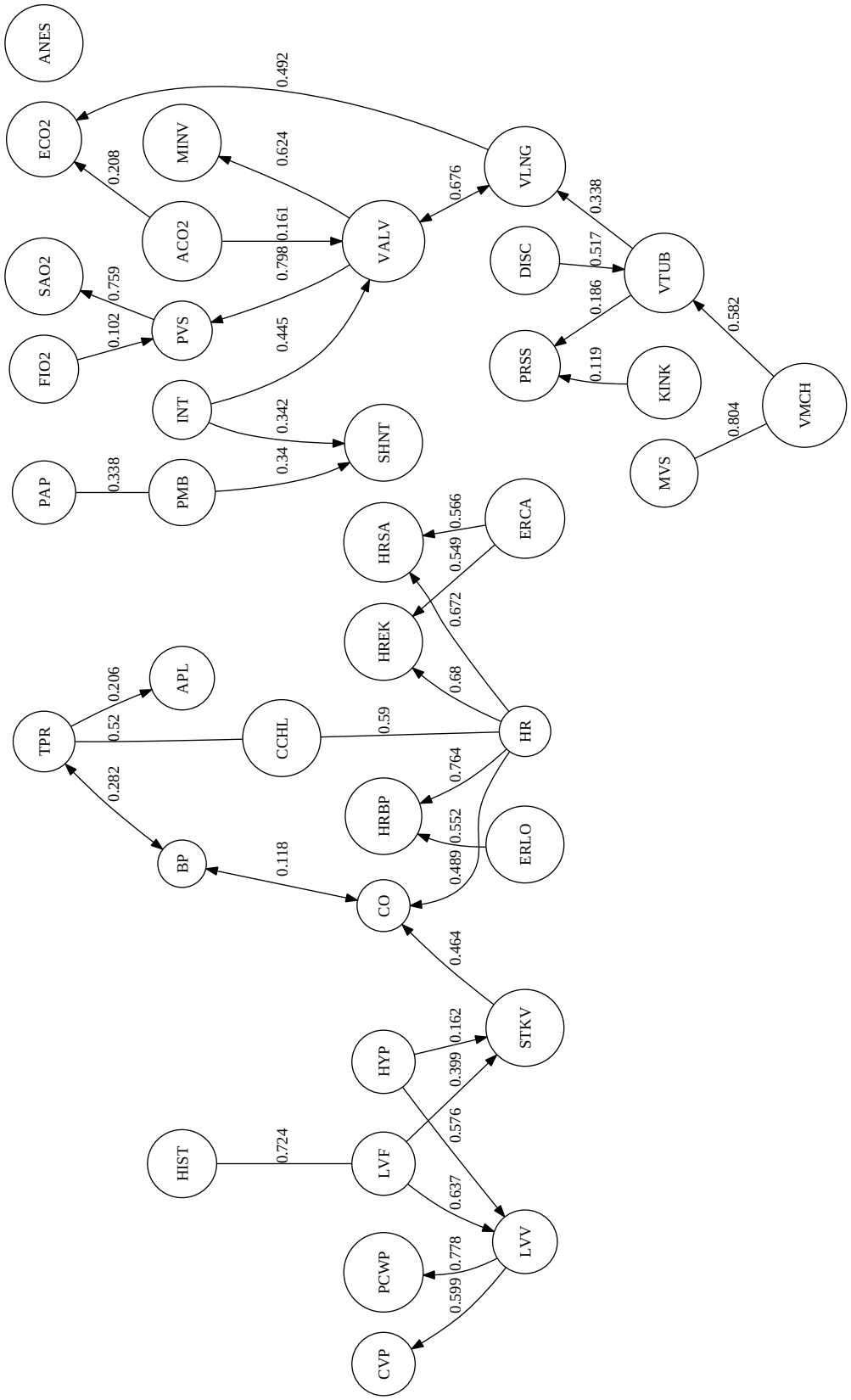


Figure 6.7: ALARM Network Learnt (Bnlearn data) by LUMIN with NMI 0.1 and NCMI 0.9 with Heuristic Link Removal

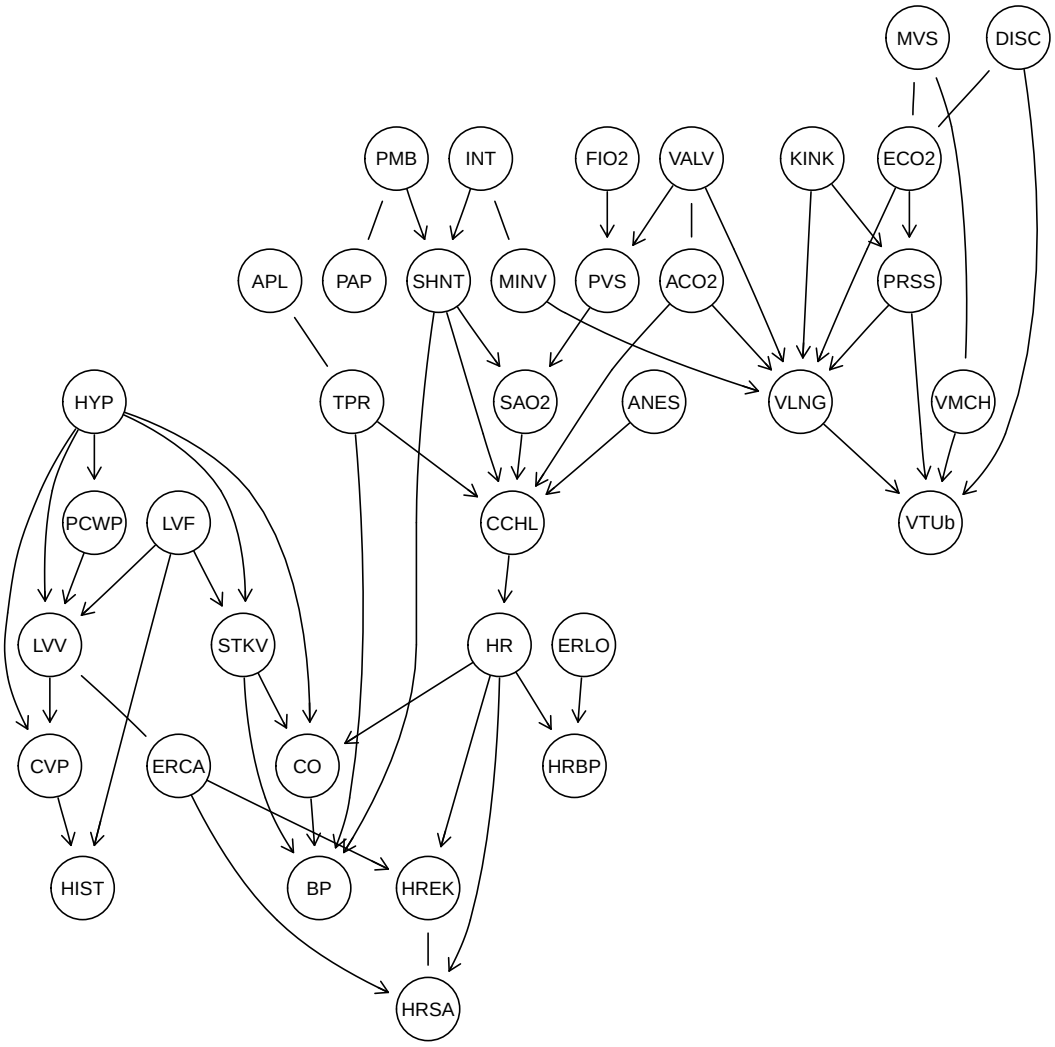


Figure 6.8: ALARM Network (Jie Chang Data) Learnt by Grow-Shrink Learner - $\alpha = 0.05$

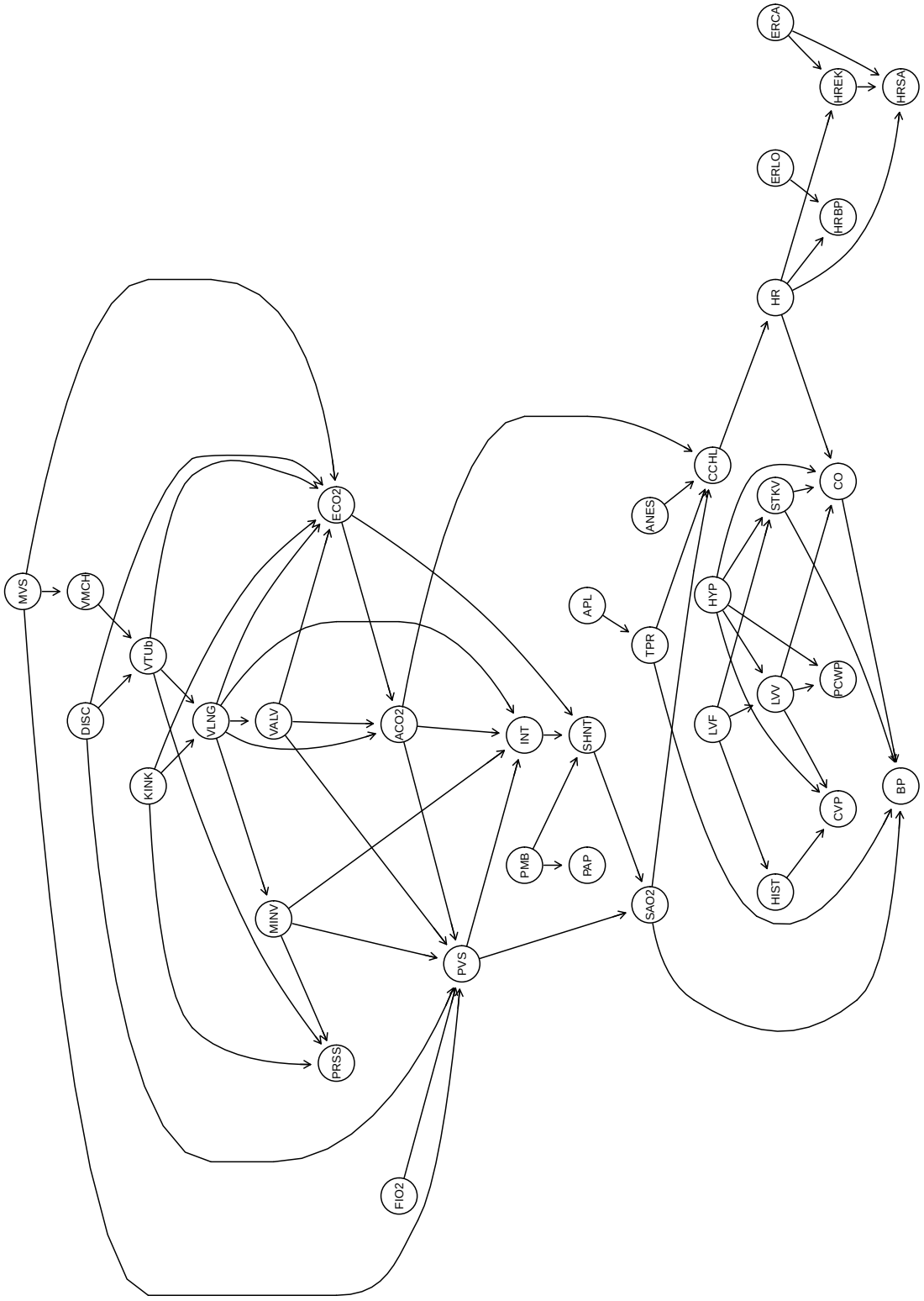


Figure 6.9: ALARM Network (Jie Chang Data) Learnt by Max-Min Hill Climbing Learner - $\alpha = 0.05$

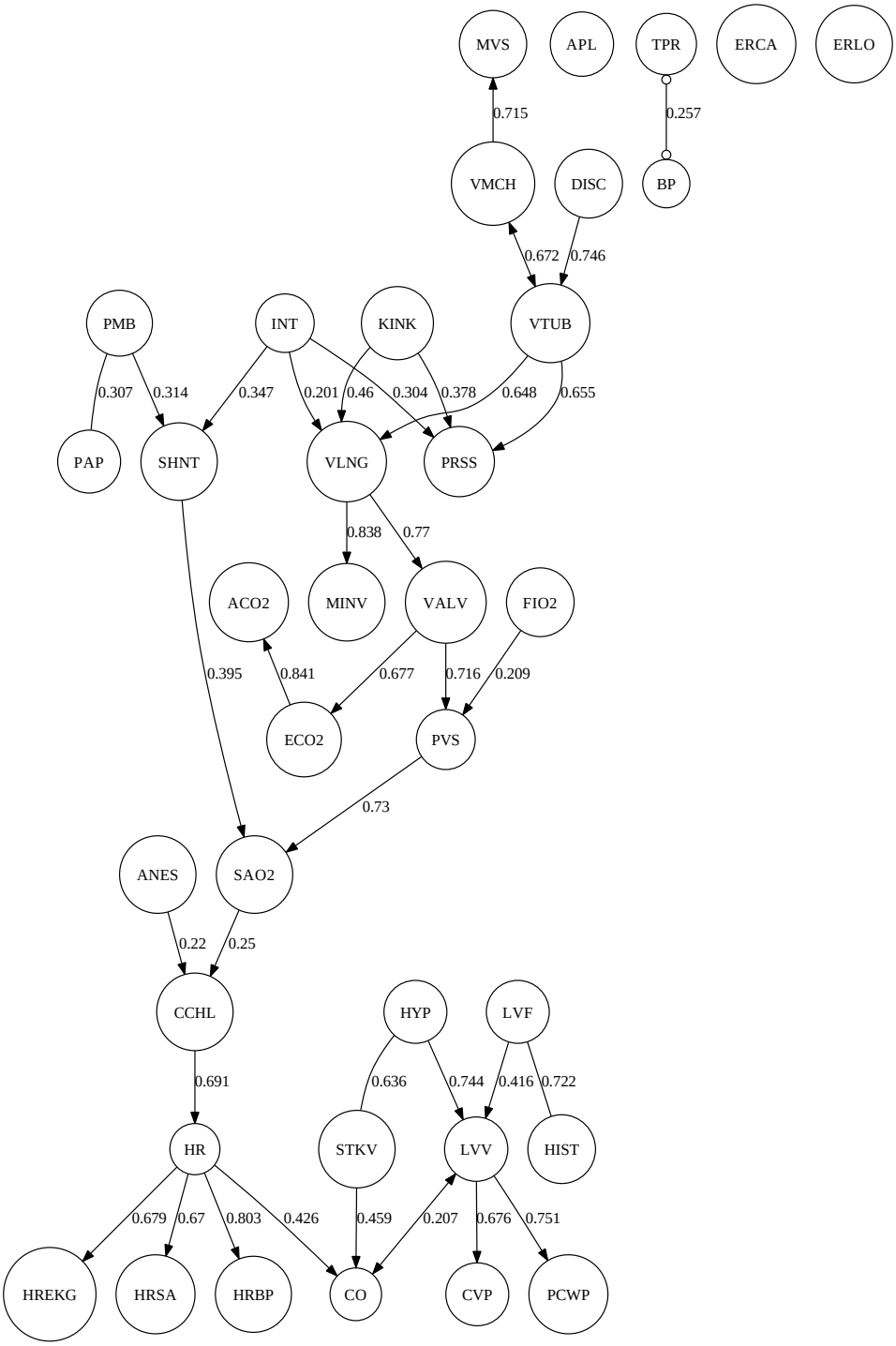


Figure 6.10: ALARM Network (Jie Chang Data) Learnt by LUMIN with NMI 0.2 and NCMI 0.5 with Heuristic Link Removal

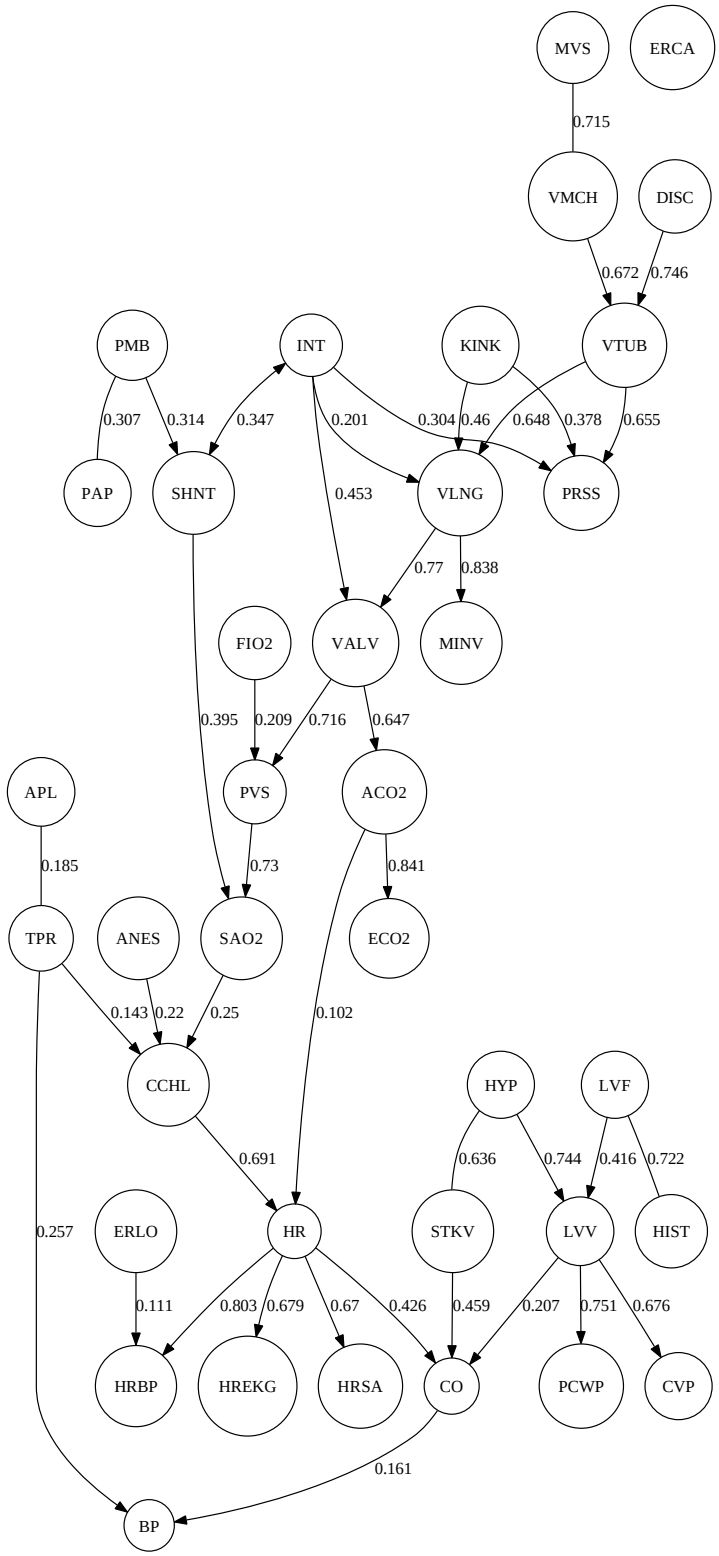


Figure 6.11: ALARM Network (Jie Chang Data) Learnt by LUMIN with NMI 0.1 and NCMI 0.9 with Heuristic Link Removal

The node *ERCA* is isolated, but all other nodes are connected and there are a total of 42 links.

6.3.6 Summary of the Results for the ALARM Network Data

Each of the learners produce a network showing the apparent relationships between the variables in the datasets. Using the CL and Hamming metrics shown in table 6.1 we can score the various learnt ALARM networks, rather than use the total score, for simplicity, we will show points lost. So, in table 6.4 lower values are better.

		ALARM (Bnlearn)		ALARM (Jie Cheng)	
		CL Metric	Hamming Distance	CL Metric	Hamming Distance
Grow-Shrink	$\alpha = 0.05$	59	35	79	49
	$\alpha = 0.1$	86	–	98	–
Hill Climbing		93	70	206	106
Max-Min Hill Climbing	$\alpha = 0.05$	68	50	90	54
	$\alpha = 0.1$	77	–	115	–
LUMIN	$NMI = 0.2, NCMI = 0.5$	176	–	*A Lot*	–
	$NMI = 0.2, NCMI = 0.5$ with link removal	99	54	76	34
	$NMI = 0.2, NCMI = 0.7$ with link removal	95	–	59	–
	$NMI = 0.05, NCMI = 0.9$ with link removal	80	37	40	19
	$NMI = 0.1, NCMI = 0.9$ with link removal	51	33	44	22

Table 6.4: Points Lost by Learners from ALARM Network Data (smaller CL values are better)

The table 6.4 shows that the LUMIN learner is comparable to the learners in the Bnlearn R package. Without the heuristic link removal LUMIN often performs relatively poorly. However, with heuristic link removal and choosing reasonably appropriate values for the NMI and NCMI thresholds, acceptable results can be achieved. An alternative view can help to clarify the performance of the learners.

Figures 6.12 and 6.13 show graphically how the performance of LUMIN varies on each dataset with varying values of NMI and NCMI. The graphs have the lost CL metric points as the y-axis, smaller values are better, the NMI value used as the x-axis, and the NCMI value represented as shading within the columns. Thus they display how changing the NMI and NCMI values alters the performance of the LUMIN learner. These figures show that for both datasets it was fairly easy for LUMIN to determine the related variables as increasing the value for NMI from 0.5 to 0.9 produced improving results. However, it appeared to be more difficult to deter-

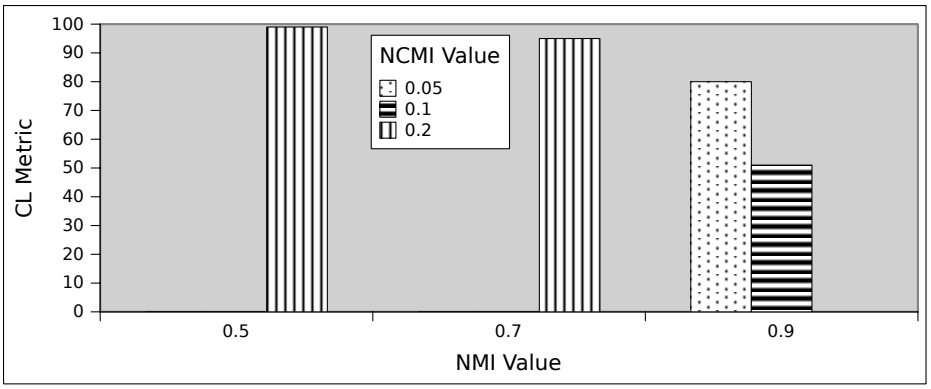


Figure 6.12: Graph of LUMIN's Performance for the ALARM Bnlearn Dataset (smaller CL values are better)

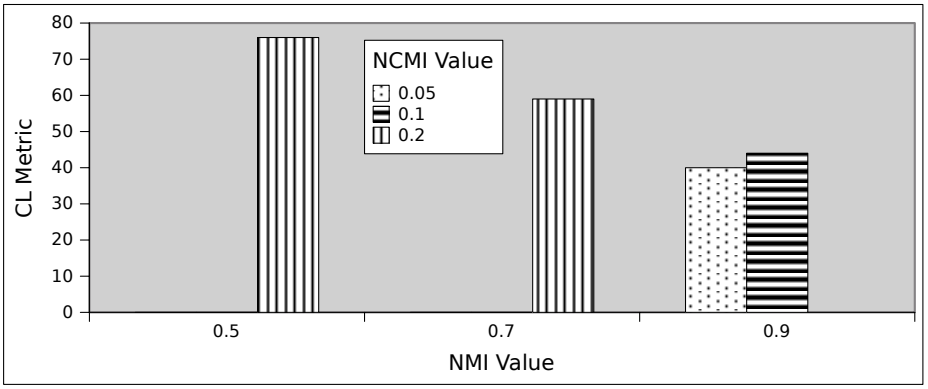


Figure 6.13: Graph of LUMIN's Performance for the ALARM Jie Chang Dataset (smaller CL values are better)

mine the direction of causality as the best results were obtained with NCMI values of 0.1 and 0.05 respectively. Figures 6.14 and 6.15 show a comparison of the performance of all the learn-

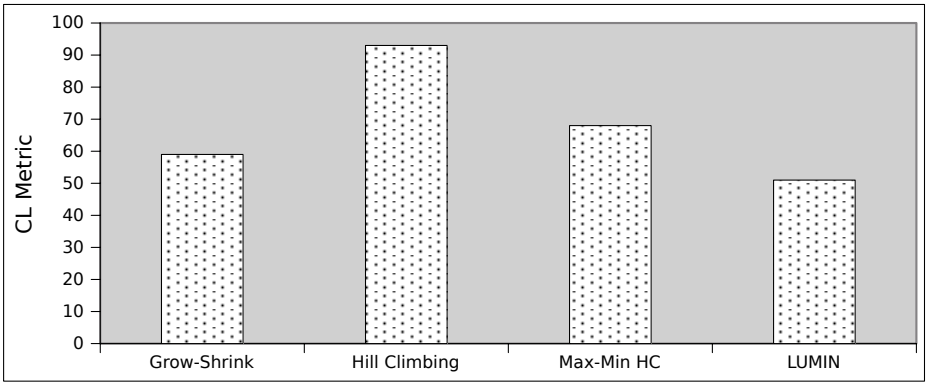


Figure 6.14: Graph of the Performance of all Learners for ALARM using the Bnlearn Dataset (smaller CL values are better)

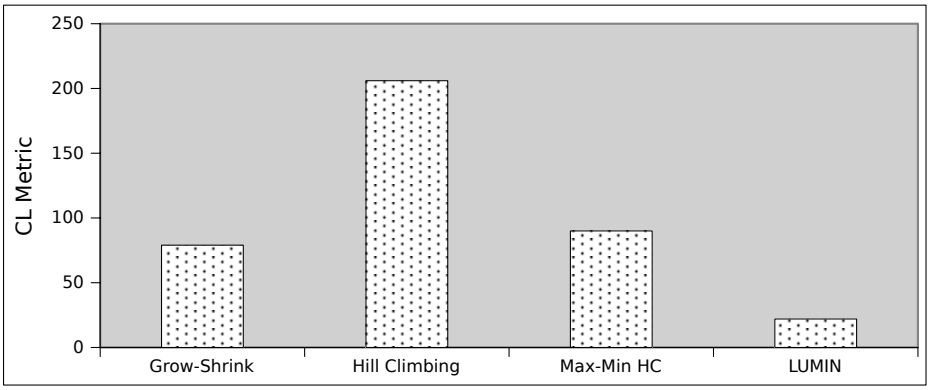


Figure 6.15: Graph of the Performance of all Learners for ALARM using the Jie Chang Dataset (smaller CL values are better)

ers on the Bnlearn and Jie Chang datasets respectively. With all the above figures a lower CL metric value indicates a better performance. Relatively LUMIN performed better on the second ALARM dataset than the first and even though both datasets are supposed to represent the same probability distribution. The best performance of the LUMIN learner occurred with different threshold values for the two datasets. We speculate that the reason for this is that the sampling method for the two datasets could be different, leading to a slightly different bias in the data. It is certainly true that the Bnlearn learners found the Bnlearn dataset much easier to process than the dataset produced by Jie Chang. However, LUMIN performed well on both datasets and was comparable to the other learners.

The ALARM network poses no particular problems being a fairly average network. The variables involved have 2, 3 or 4 possible values so there is a significant degree of variation in

the possible entropy of individual variables, this may effect the level of NMI which should be considered significant. This could be why some known causal links have very low values of NMI making finding them difficult.

6.4 Hailfinder

Hailfinder, [Abramson et al 1996], is an experimental system intended to help forecast severe weather in Northeastern Colorardo. The core of the system is a BN which was constructed with the help of existing data, expert judgement and existing understanding of some of the physical processes involved in determining the changing weather patterns. To the best of our knowledge the project has not been continued to the point where the forecasting of the Hailfinder system has been analysed for accuracy and usefulness, but the resulting network is often used as a standard in learning problems.

The Hailfinder network consists of 56 variables with 66 relationships between them. The network is shown in figure 6.16. The variables involved are outlined in table 6.5 below:

Table 6.5: Hailfinder Network Variables

Variable Name	Description	Possible Values
SubjVertMo	subjective judgement	StrongUp, WeakUp, Neutral and Down
N07muVerMo	10.7mu vertical motion of vertical motion	StrongUp, WeakUp, Neutral and Down
QGVertMotion	quasigeostrophic vertical motion	StrongUp, WeakUp, Neutral and Down
CombVerMo	combined vertical motion	StrongUp, WeakUp, Neutral and Down
AreaMesoALS	area of meso-alpha	StrongUp, WeakUp, Neutral and Down
SatContMoist	satellite contribution to moisture	VeryWet, Wet, Neutral and Dry
RaoContMoist	reading at the forecast center for moisture	VeryWet, Wet, Neutral and Dry
CombMoisture	combined moisture	VeryWet, Wet, Neutral and Dry
AreaMoDryAir	area of moisture and dry air	VeryWet, Wet, Neutral and Dry
VISCloudCov	visible cloud cover	Cloudy, PC and Clear
IRCloudCover	infrared cloud cover	Cloudy, PC and Clear
CombClouds	combined cloud cover	Cloudy, PC and Clear
CldShadeOth	cloud shading, other	Cloudy, PC and Clear
AMInstabMt	AM instability in the mountains	None, Weak and Strong
InsInMt	instability in the mountains	None, Weak and Strong
WndHodograph	wind hodograph	DCVZFavor, StrongWest, Westerly and Other

Variable Name	Description	Possible Values
SubjVertMo	subjective judgement	StrongUp, WeakUp, Neutral and Down
OutflowFrMt	outflow from mountains	None, Weak and Strong
MorningBound	morning boundaries	None, Weak and Strong
Boundaries	boundaries	None, Weak and Strong
CldShadeConv	cloud shading, convection	None, Some and Marked
CompPlFcst	composite plains forecast	IncCapDecIns, LittleChange and DecCapIncIns
CapChange	capping change	Decreasing, LittleChange and Increasing
LoLevMoistAd	low-level moisture advection	StrongPos, WeakPos, Neutral and Negative
InsChange	instability change	Decreasing, LittleChange and Increasing
MountainFcst	mountains (region 1) forecast	XNIL, SIG and SVR
Date	date	May15_Jun14, Jun15_Jul1, Jul2_Jul15, Jul16_Aug10, Aug11_Aug20 and Aug20_Sep15
Scenario	scenario	A, B, C, D, E, F, G, H, I, J and K
ScenRelAMCIN	scenario relevant to AM convective inhibition	AB and C to K
MorningCIN	morning convective inhibition	None, PartInhibit, Stifling and TotalInhibit
AMCINInScen	AM convective inhibition in scenario	LessThanAve, Average and MoreThanAve
CapInScen	capping withing scenario	LessThanAve, Average and MoreThanAve
ScenRelAMIns	scenario relevant to AM instability	ABI, CDEJ, F, G, H and K
LIfr12ZDENSd	LI from 12Z DEN sounding	LIGt0, N1GtLIGt_4, N5GtLIGt_8 and LILt_8
AMDewptCalPl	AM dewpoint calculations, plains	Instability, Neutral and Stability
AMInsWliScen	AM instability within scenario	LessUnstable, Average and MoreUnstable
InsSclInScen	instability scaling within scenario	LessUnstable, Average and MoreUnstable
ScenRel34	scenario relevant to regions 2/3/4	ACEFK, B, D, GJ and HI
LatestCIN	latest convective inhibition	None, PartInhibit, Stifling and TotalInhibit
LLIW	LLIW severe weather index	Unfavorable, Weak, Moderate and Strong
CurPropConv	current propensity to convection	None, Slight, Moderate and Strong
ScnRelPlFcst	scenario relevant to plains forecast	A, B, C, D, E, F, G, H, I, J and K

Variable Name	Description	Possible Values
SubjVertMo	subjective judgement	StrongUp, WeakUp, Neutral and Down
PlainsFcst	plains forecast	XNIL, SIG and SVR
N34StarFcst	regions 2/3/4 forecast	XNIL, SIG and SVR
R5Fcst	region 5 forecast	XNIL, SIG and SVR
Dewpoints	dewpoints	LowEverywhere, LowAtStation, LowSHighN, LowNHighS, LowMtsHighPl, HighEverywher, Other
LowLLapse	low-level lapse rate	CloseToDryAd, Steep, ModerateOrLe and Stable
MeanRH	mean relative humidity	VeryMoist, Average and Dry
MidLLapse	mid-level lapse rate	CloseToDryAd, Steep and ModerateOrLe
MvmtFeatures	movement of features	StrongFront, MarkedUpper, OtherRapid and NoMajor
RHRatio	relative humidity ratio	MoistMDryL, DryMMoistL and other
SfcWndShfDis	surface wind shifts and discontinuities	DenvCyclone, E_W_N, E_W_S, MovigFtorOt, DryLine, None and Other
SynForcng	synoptic forcing	SigNegative, NegToPos, SigPositive, PosToNeg and LittleChange
TempDis	temperature discontinuities	QStationary, Moving, None, Other
WindAloft	wind aloft	LV, SWQuad, NWQuad, AllElse
WindFieldMt	wind fields, mountains	Westerly and LVorOther
WindFieldPln	wind fields, plains	LV, DenvCyclone, LongAnticyc, E_NE, SEquad and WidespdDnsl

Like the ALARM network the data variables are categorical, that is they can only take on a small number of fixed values as indicated above.

6.4.1 Results

As with the ALARM datasets we show the graphs learnt by various learners and discuss some of the issues they encounter. The same data was analysed using machine learners from Bnlearn and also using LUMIN. We used datasets generated by the Bnlearn package and supplied by

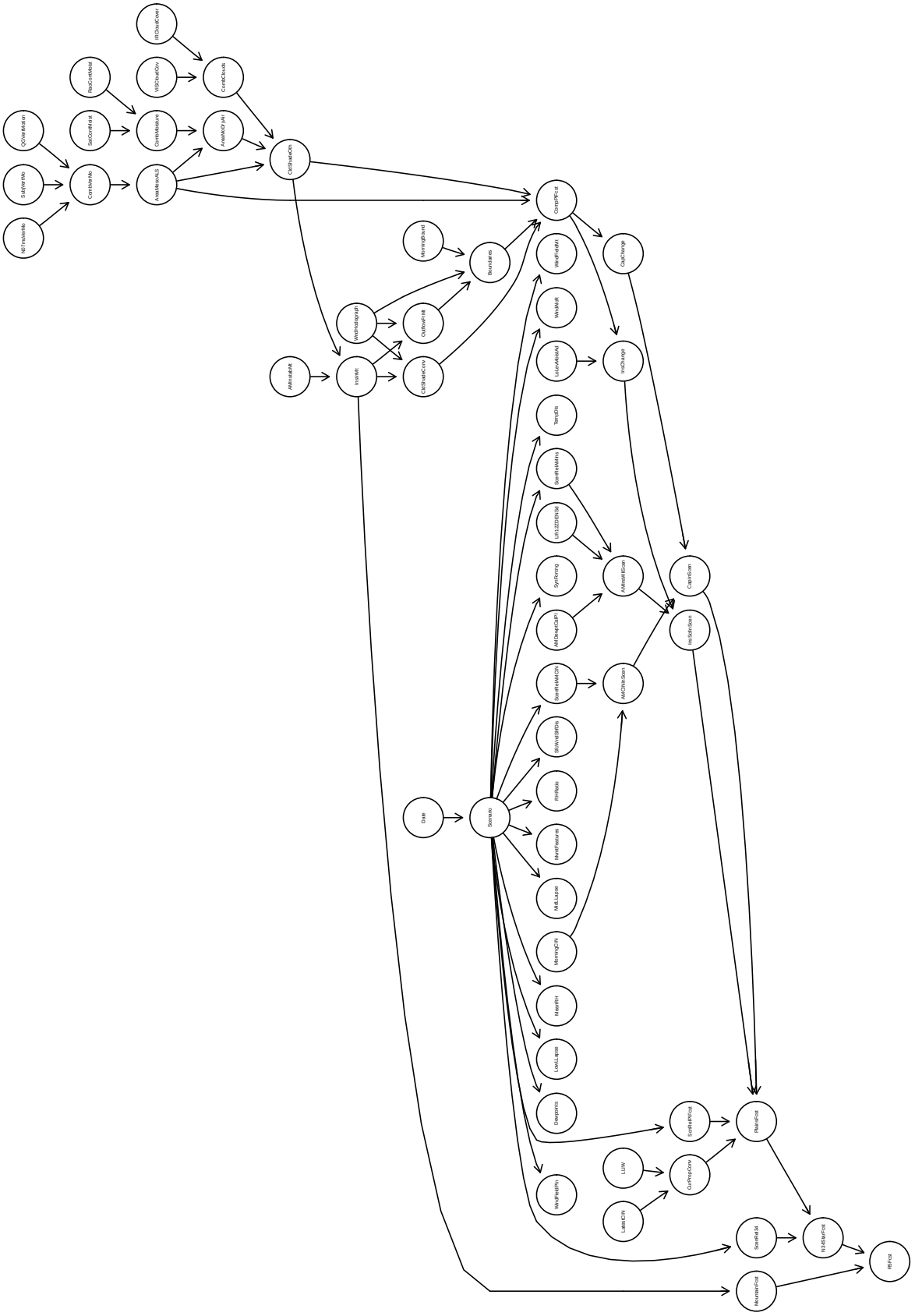


Figure 6.16: The Hailfinder Network

the Discovery Systems Laboratory[10], DSL[11]. We combined four of the 5000 record datasets to make a 20000 record dataset. The information learnt is described below. We examine areas of the graphs which caused problems for the LUMIN learner and discuss how altering its NMI and NCMI thresholds varies what is learnt. We also introduce LUMIN's domain ranking feature and show how it can be used to help clarify relationships.

6.4.2 Bnlearn R Package - Bnlearn Data

As before there are a number of learners supplied by the Bnlearn R package. The network obtained, using the optimised Grow-Shrink Markov Blanket learner with a target nominal type I error rate of 0.05 is shown in figure 6.17. This network has 80 relations between the variables and it has become split into 5 separate networks with a single additional isolated node. The learner notes that there is a problem with the v-structure $AMCINInScen \rightarrow CapInScen \leftarrow InsChange$ as one or both of the links were already assigned a different direction. If the α level is changed to 0.1 the Grow-Shrink learner produces the network shown in figure B.22,which is still in 5 separate networks with a single additional node, but now has 83 links. In learning this network the learner noted that the v-structure $CapInScen \rightarrow InsChange \leftarrow LoLevMoistAd$ has one or both of its links already assigned a different direction.

The Hill-Climbing score-based algorithm produces a graph shown in figure 6.18 that has 64 links. The Max-Min Hill Climbing learner with $\alpha = 0.05$ produces the graph shown in figure 6.19. This consists of three separate networks having a total of 64 links with no isolated nodes. Using the same learner with $\alpha = 0.1$ the graphs shown in figure B.11 is produced. This is identical to figure 6.19 so in this instance the change in the value of alpha, that is, doubling the target nominal type I error rate of the conditional independence test, made no different to the learnt graph.

6.4.3 LUMIN Learner - Bnlearn Data

The Hailfinder network has aspects which are particularly difficult for a local constraint-based learner like LUMIN, many nodes share a single parent, which makes it hard to distinguish the effect of the child from the parent, and a large number of nodes in general have only a single

[10]The data we used was originally generated for paper [Tsamardinos et al 2006]. Information about DSL is available online at `http://www.dsl-lab.org/`.

[11]The dataset used can be obtained from `http://www.dsl-lab.org/supplements/mmhc_paper/hailf_data.zip`.

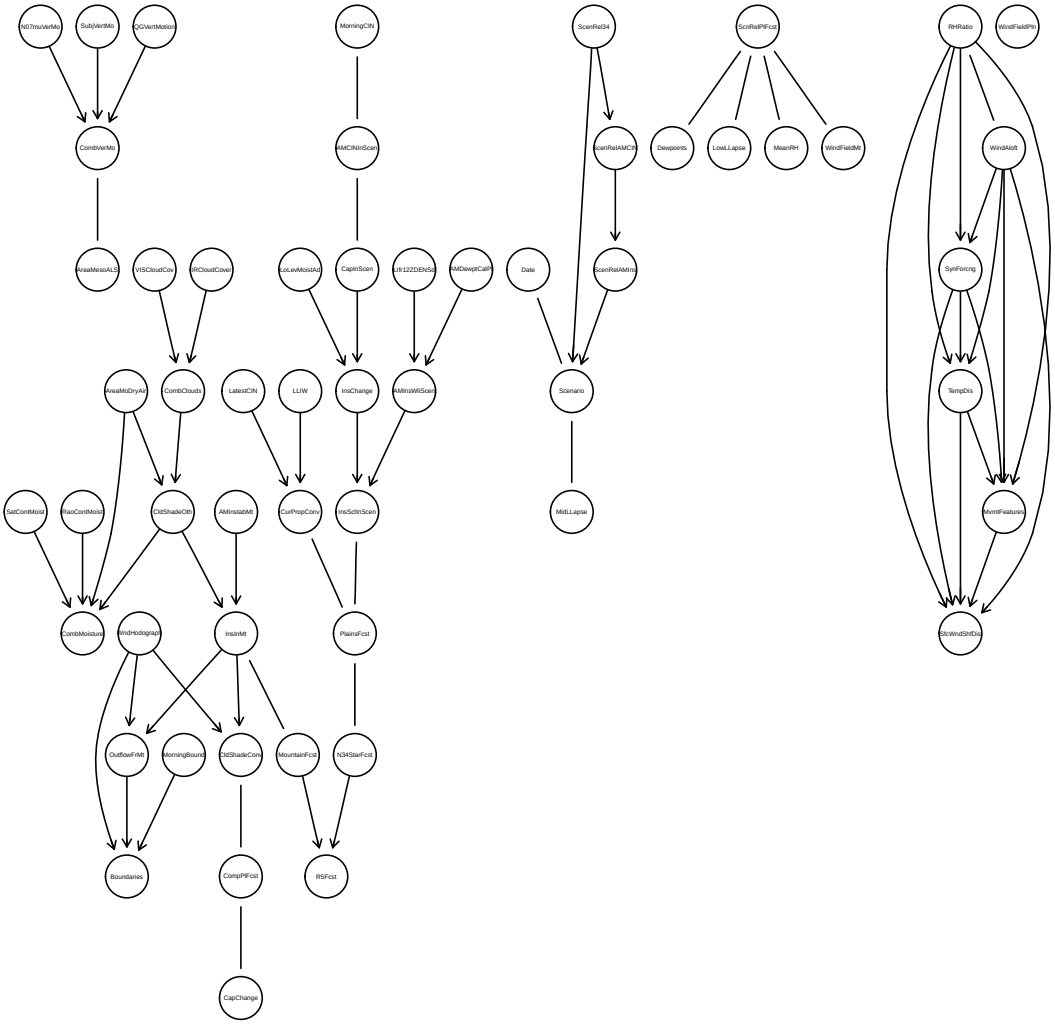


Figure 6.17: Hailfinder Network Learnt by the Grow-Shrink Learner with $\alpha = 0.05$

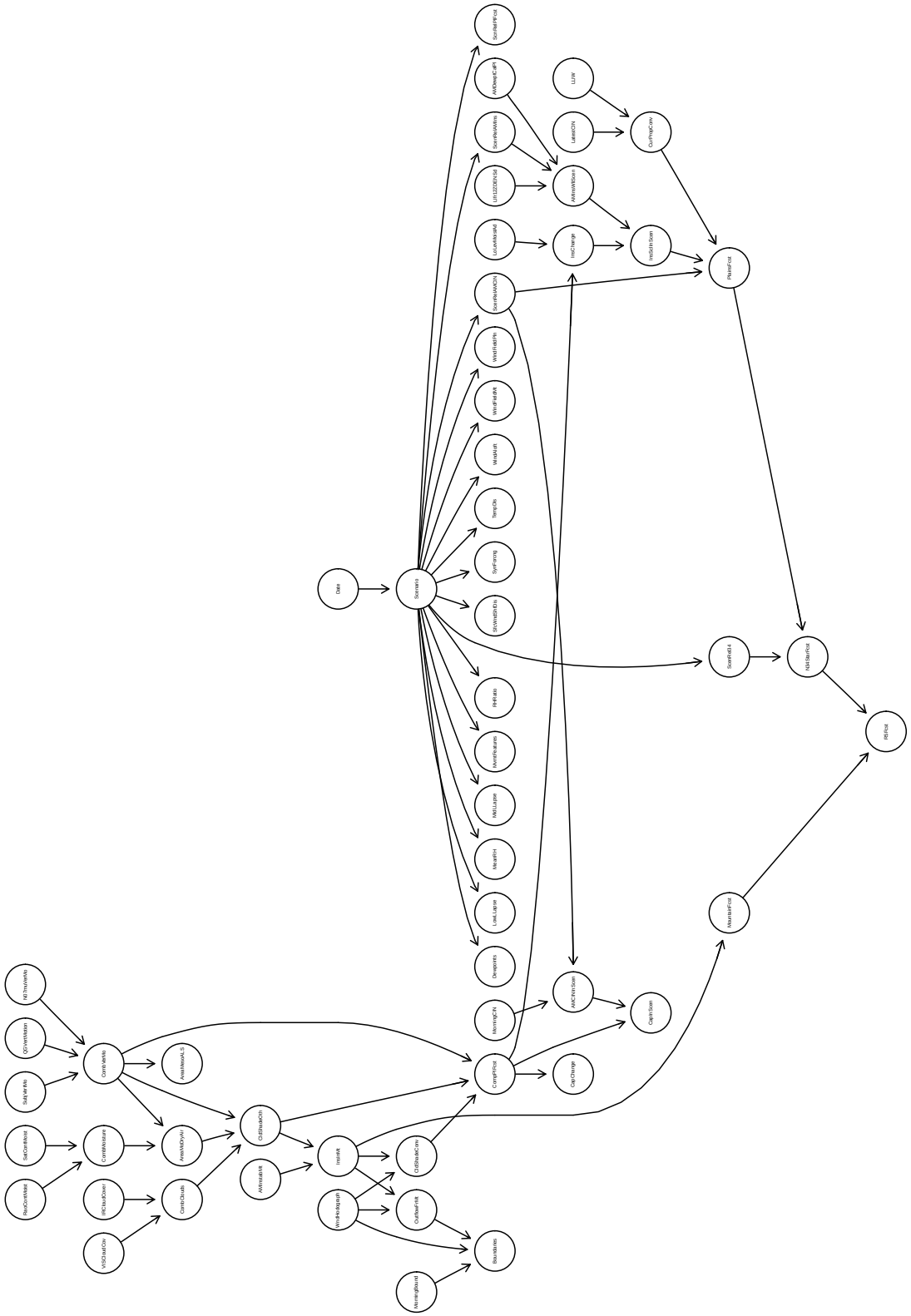


Figure 6.18: Hailfinder Network Learnt by the Hill-Climbing Learner

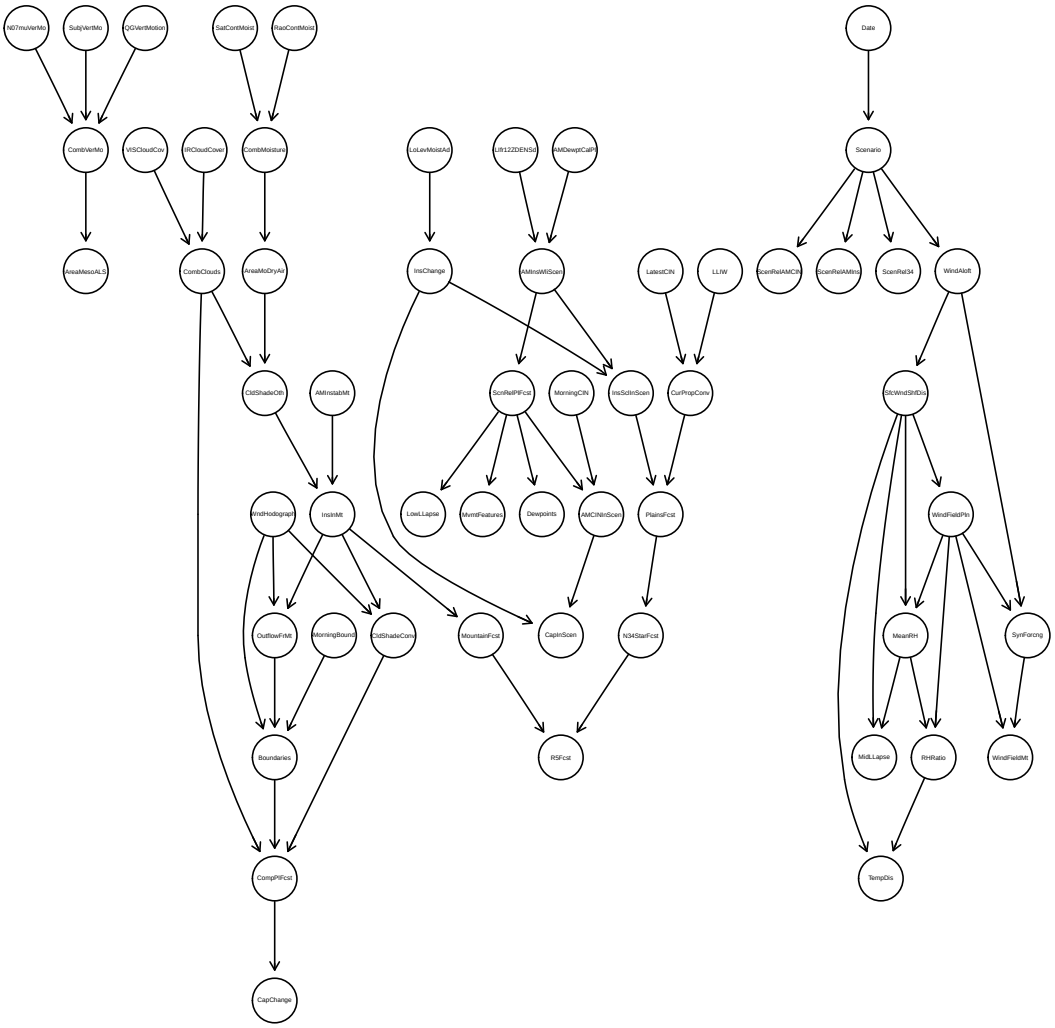


Figure 6.19: Hailfinder Network Learnt by the Max-Min Hill Climbing Learner with $\alpha = 0.05$

parent, which can make determining the direction of influence difficult. Figure B.12 shows the graph learnt with a NMI threshold of 0.02 a NCMI threshold of 0.5 and no heuristic link removal. There were a total of 140 triplets that gave rise to contradictory link directions, they are listed in Appendix C. That is a considerable number of conflicted triplets, part of the problem is that with so many closely related nodes there are inevitably many incorrect links in the initial graph which leads to uncertainty in their direction.

One method of helping with this problem is to remove one of a pair of very closely related nodes. Since currently we have no method of choosing 'the best' nodes to delete, and deleting any node will distort the network, this will inevitably cause a reduction in the 'score' of any learnt network, but can be useful in clarifying parts of the network with less clutter. Figure B.20 shows the Hailfinder network learnt with the same thresholds as figure B.12, but allowing one of a pair of closely related nodes to be deleted. In the case of this graph of the pair of nodes *CombVerMo* and *AreaMesoALS* the node *CombVerMo* was removed, and of *CompPlFcst* and *CapChange* the node *CompPlFcst* was removed and lastly of the nodes *Scenario* and *ScnRelPlFcst* the node *Scenario* was removed. This lead to a reduction in links from 189 to 156, and a reduction in reversed links, those which end up bi-directed, from 63 to 47. Figures 6.20 and B.13 show the graphs learnt with the same thresholds both with heuristic link removal and in the second figure also allowing similar node removal. These graphs have 80 and 59 links respectively of which 63 and 47 are bi-directed.

It is interesting to note that the number of bi-directed links is not effected by the heuristic link removal suggesting that the bi-directed links are not obviously an artifact and may be a valid part of the graphs. One possibility here would be to effectively rerun link direction determination after the heuristic link removal, as superfluous links can cause problems in direction determination. As with the ALARM dataset it can be useful to examine what is learnt with different threshold values and so we have figure B.15 which shows the graph learnt with an NMI threshold of 0.02 and an NCMI threshold of 0.9 and figure 6.21 which is the graph learnt with an NMI threshold of 0.01 and an NCMI threshold of 0.9. The graphs show that in the Hailfinder dataset a number of variables are very closely related, so closely in fact that their normalised mutual information has a value close to 1. The effect of this is that it is not possible for LUMIN to determine any difference between such variables, their causes and effects will appear to be shared. Heuristic

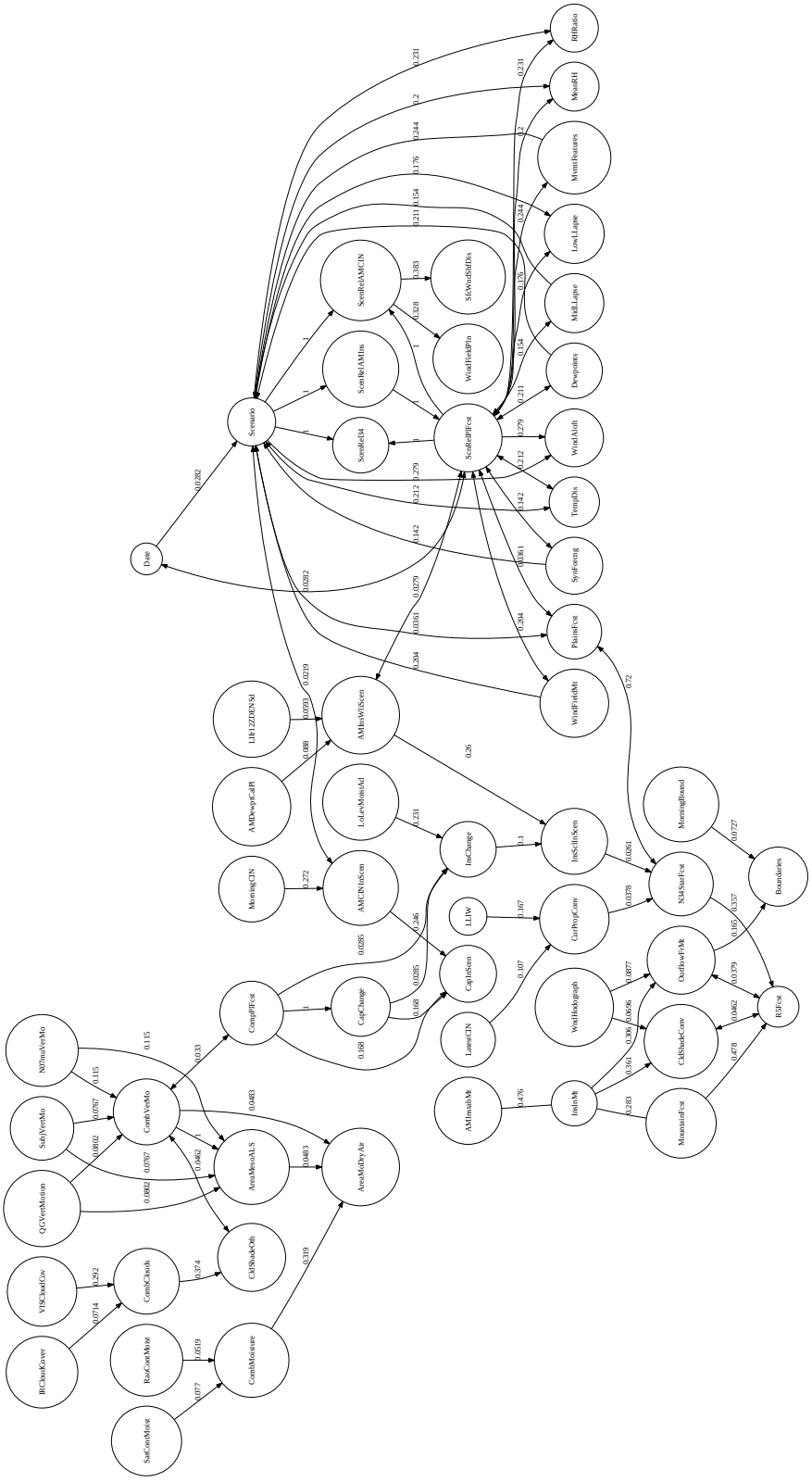


Figure 6.20: The Hailfinder Network Learnt by LUMIN with NMI of 0.02 and NCMI 0.5 with Heuristic Link Removal

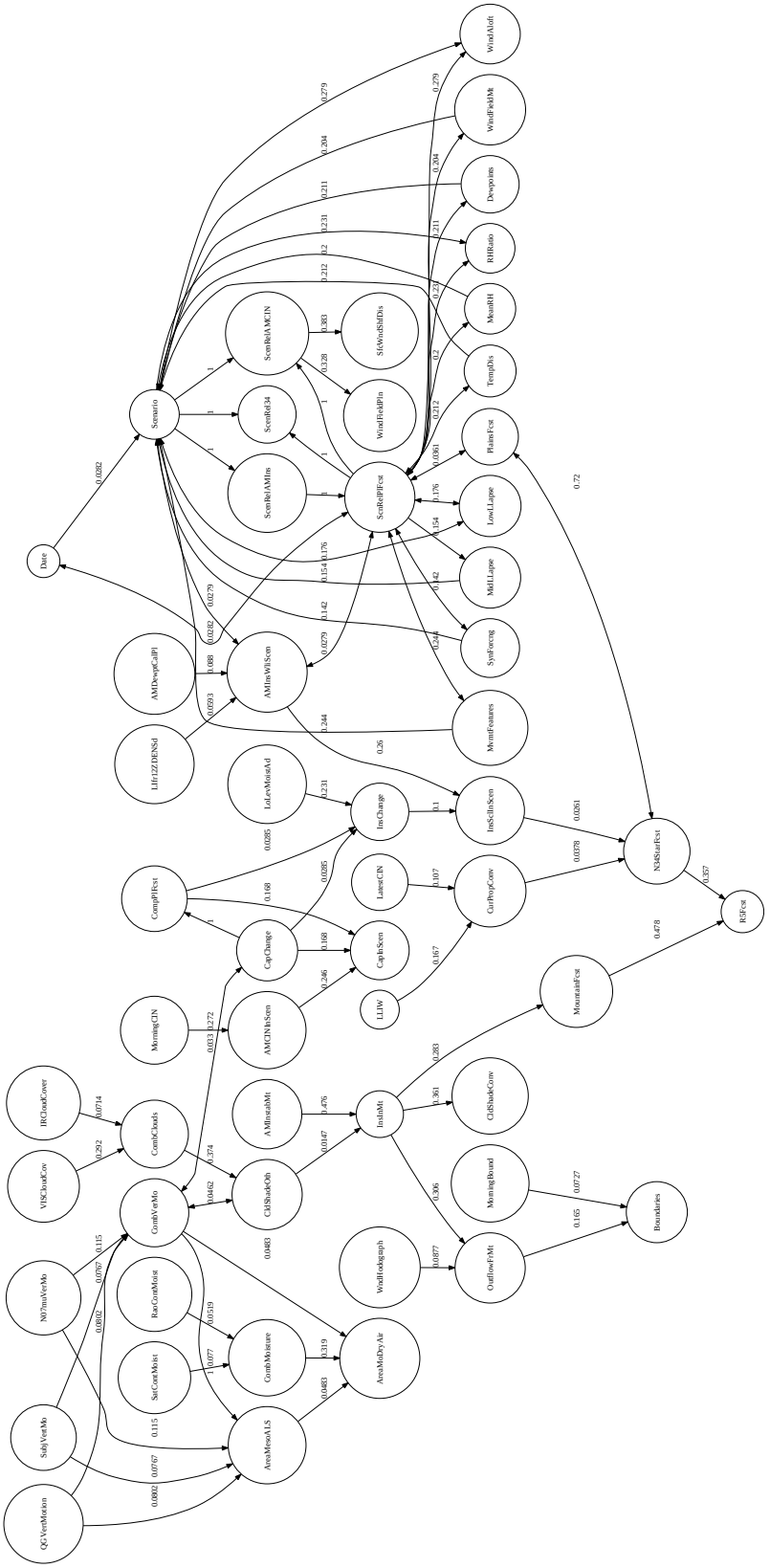


Figure 6.21: The Hailfinder Network Learnt by LUMIN with NMI of 0.01 and NCMI 0.9 with Heuristic Link Removal

link removal will help reduce multiple links, but is not able to sensibly differentiate such nodes so the link removal may become rather arbitrary.

At this point we have an option to attempt to aid the determination of direction using the domain feature of LUMIN. This is similar to, but a lot more flexible than the node ordering used in learners like K2. Node ordering usually specifies a unique direction between every node. The domain feature of LUMIN allows nodes to be arbitrarily grouped into domains and to be assigned a ranking within those domains. A node can belong to many domains, when the link between two nodes is being examined for direction the first thing that is done is to see if the nodes share any domains and if so if their rankings differ. A low ranking node is said to be a cause of higher ranking nodes within the same domain. The link direction chosen by this mechanism is sacrosanct, and not effected by the usual link direction determination based on NCMI thresholds. We have avoided use of the domain feature so far as specifying domain information makes the comparison of LUMIN with other learners difficult as they do not include a similar feature. However, purely for interest we have divided the nodes in the Hailfinder network into three groups within a single domain with ranking of 10, 15 and 20. The groups are shown in table 6.6.

Ranking 10	Ranking 15	Ranking 20
N07muVerMo	CompPlFcst	CapChange
SubjVertMo	LoLevMoistAd	InsChange
QGVertMotion	ScenRelAMCIN	MountainFcst
CombVerMo	MorningCIN	AMCINInScen
AreaMesoALS	MeanRH	CapInScen
SatContMoist	ScenRelAMIns	AMInsWliScen
RaoContMoist	LIfr12ZDENSd	InsSclInScen
CombMoisture	AMDewptCalPl	ScenRel34
AreaMoDryAir	Dewpoints	LatestCIN
VISCloudCov	LowLLapse	LLIW
IRCloudCover	MidLLapse	CurPropConv
CombClouds	MvmtFeatures	ScnRelPlFcst
CldShadeOth	RHRatio	PlainsFcst
AMInstabMt	SfcWndShfDis	N34StarFcst
InsInMt	SynForcng	R5Fcst
WndHodograph	TempDis	
OutflowFrMt	WindAloft	
MorningBound	WindFieldMt	
Boundaries	WindFieldPln	
CldShadeConv		
Date		
Scenario		

Table 6.6: Hailfinder Variables in Ranking Groups

Using this ranking with an NMI threshold of 0.02 a NCMI threshold of 0.5 and allowing heuristic link removal produces the graph shown in figure B.26. Similarly figure B.16 shows the graph learnt by LUMIN with an NMI threshold of 0.02 and an NCMI threshold of 0.7, figure B.17 shows the graph learnt using the same thresholds, but with the addition of the domain information. Lastly figures B.14 and B.18 show the graphs learnt with a NCMI threshold of 0.9 and NMI thresholds of 0.01 and 0.02 respectively with the use of domain information.

6.4.4 Bnlearn R Package - DSL Data

The network obtained, using the optimised Grow-Shrink Markov Blanket learner with a target nominal type I error rate of 0.05 is shown in figure 6.22. This network has 85 relations between the variables and it has become split into 7 separate networks with a single additional isolated node. The learner notes that there is a problem with the v-structures TempDis $\rightarrow$ WindAloft $\leftarrow$ WindFieldMt, TempDis $\rightarrow$ SynForcng $\leftarrow$ WindFieldMt, MvmtFeatures $\rightarrow$ WindAloft $\leftarrow$ WindFieldMt, TempDis $\rightarrow$ SfcWndShfDis $\leftarrow$ WindFieldMt, and CldShadeOth $\rightarrow$ AreaMoDryAir $\leftarrow$ CombMoisture as one or both of the links were already assigned a different direction. If the α level is changed to 0.1 the Grow-Shrink learner produces the network shown in figure B.19, which is still in 5 separate networks with a single additional node, but now has 93 links. In learning this network the learner noted that the v-structure $MvmtFeatures \rightarrow WindAloft \leftarrow WindFieldMt$ has one or both of its links already assigned a different direction.

The Hill-Climbing score-based algorithm produces a graph shown in figure 6.23 that has 65 links. The Max-Min Hill Climbing learner with $\alpha = 0.05$ produces the graph shown in figure 6.24. This consists of two separate networks and an isolated node having a total of 67 links. Using the same learner with $\alpha = 0.1$ the graphs shown in figure B.23 is produced. This graph has two networks with no isolated nodes and a total of 70 links. So on this dataset unlike the other Hailfinder dataset changing the value of alpha from 0.05 to 0.1 did make a difference to the graph learnt by the Max-Min Hill Climbing learner.

6.4.5 LUMIN Learner - DSL Data

Figure B.25 shows the graph learnt with a NMI threshold of 0.02 a NCMI threshold of 0.5 and no heuristic link removal. The result is a single network with 189 links and no isolated nodes. As with the Bnlearn Hailfinder data there were a considerable number of contradictory direction indicators giving rise to 56 reversed links. For comparison we include figure B.21 which

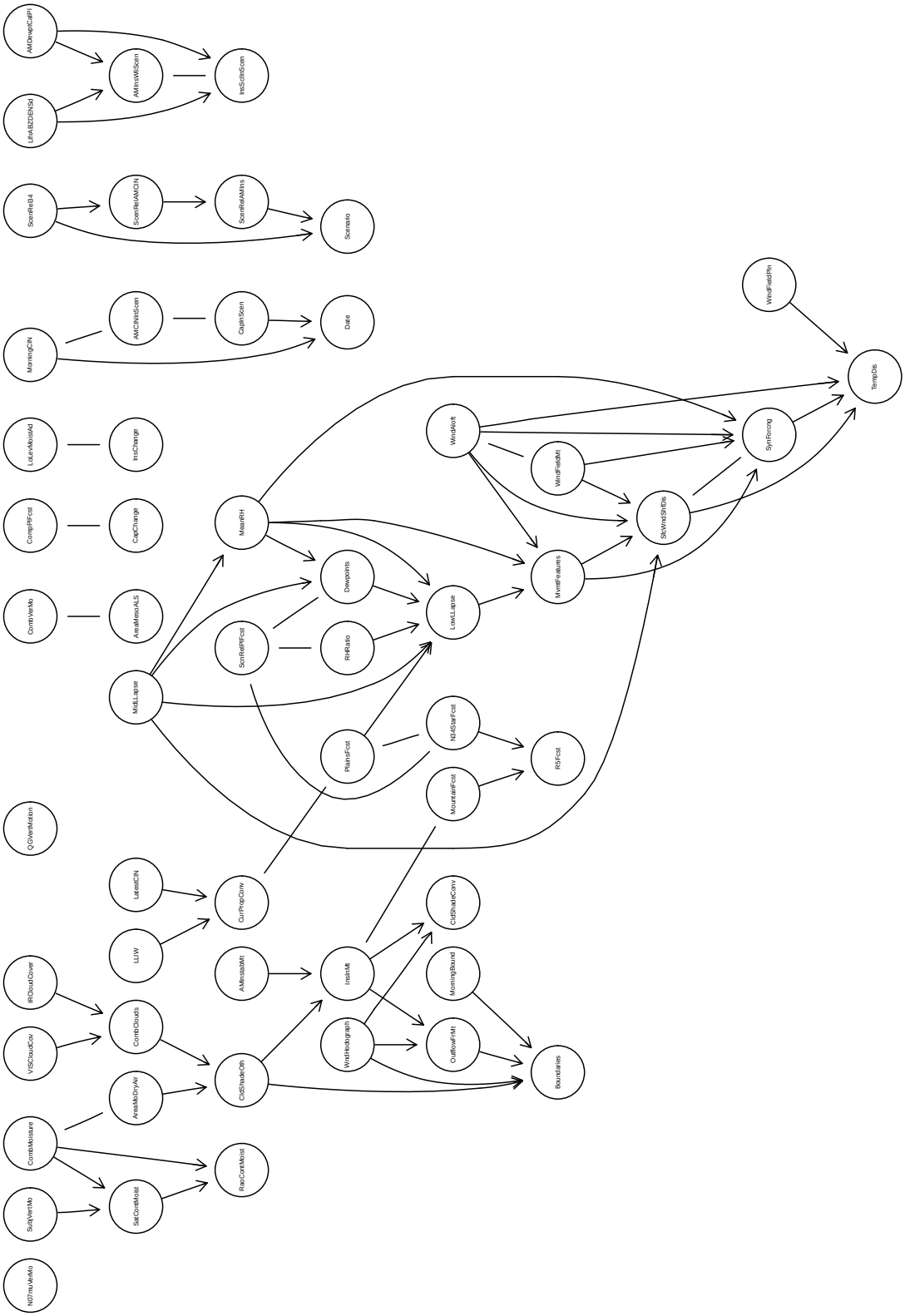


Figure 6.22: Hailfinder Network Learnt by the Grow-Shrink Learner with $\alpha = 0.05$

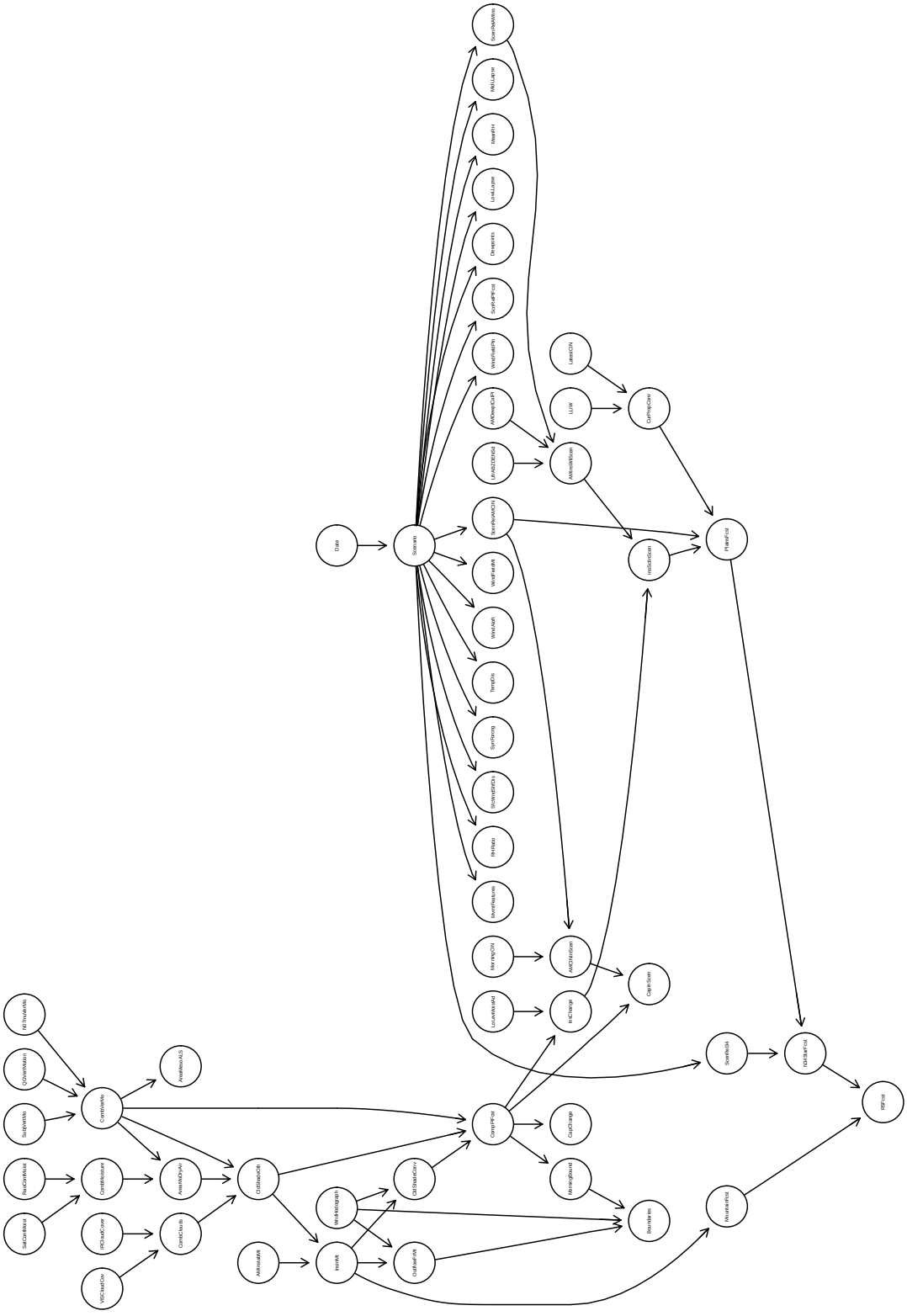


Figure 6.23: Hailfinder Network Learnt by the Hill-Climbing Learner

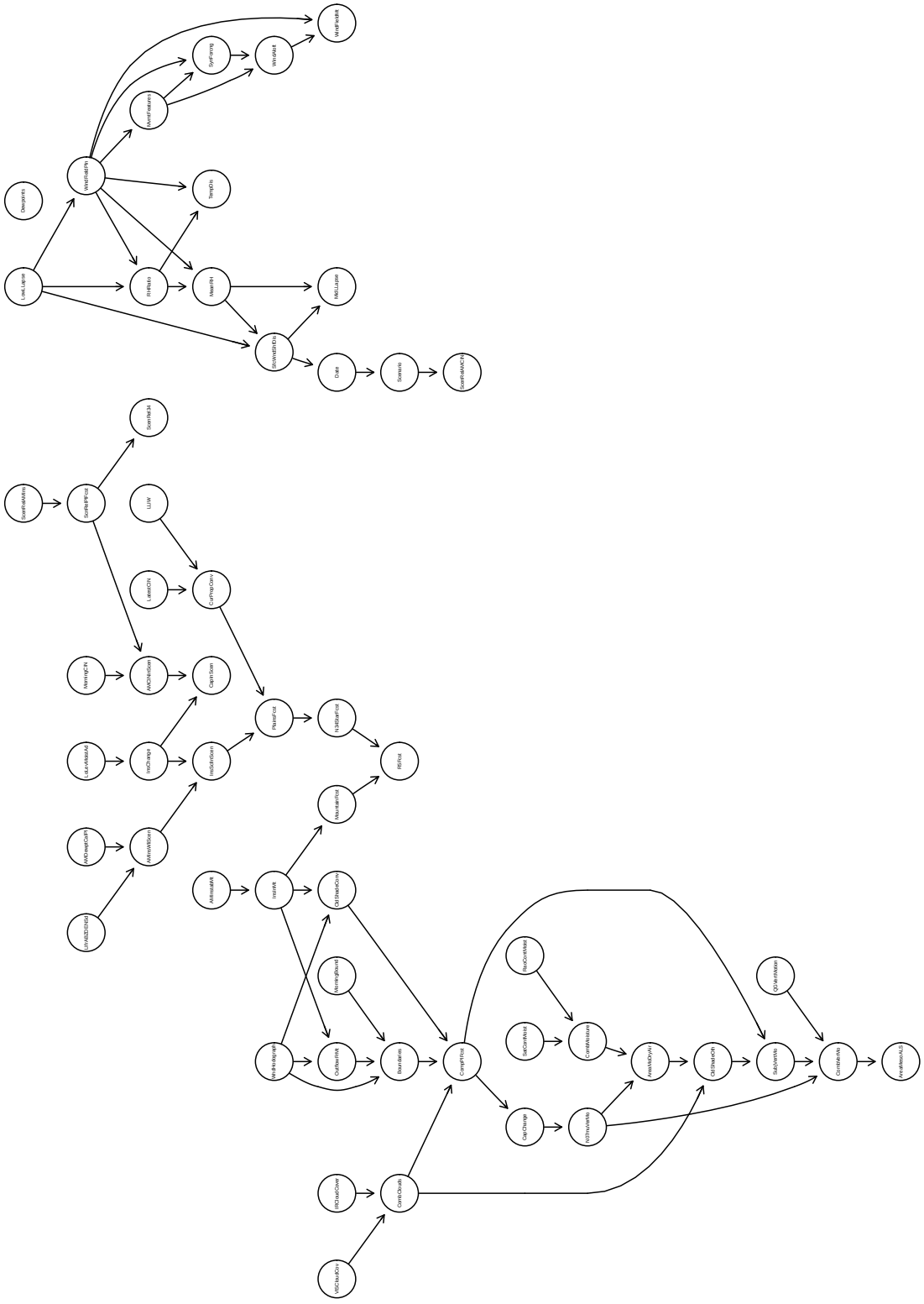


Figure 6.24: Hailfinder Network Learnt by the Max-Min Hill Climbing Learner with $\alpha = 0.05$

shows the graphs learnt using the same thresholds, but with the addition of the same domain information used with the Bnlearn Hailfinder data above. This graph has 78 links of which 23 are bidirectional. Figure B.27 shows the graph learnt by LUMIN with thresholds of 0.02 for NMI and 0.5 for NCMI and using heuristic link removal. Figure B.28 shows the graph learnt by LUMIN with thresholds of 0.02 for NMI and 0.7 for NCMI using heuristic link removal. The graph has no isolated nodes and 78 links. Figures 6.25 and B.29 show the graphs learnt by LUMIN with an NMI thresholds of 0.01 and 0.02 respectively with an NCMI threshold of 0.9. The graphs have 77 and 82 links, and no isolated nodes. Lastly we have figures B.30 and B.31 which show the graphs learnt with an NMI threshold of 0.02 and NCMI thresholds of 0.7 and 0.9 respectively and figure B.31 which has thresholds of 0.01 and 0.9 for NMI and NCMI respectively, all these graphs were generated using domain information.

6.4.6 Summary of the Results for the Hailfinder Network Data

We have analysed some of the graphs produced by each of the learners for both datasets using the CL metric, the result is shown in table 6.7. The Hill Climbing learner outperformed all other

		Hailfinder (Bnlearn) CL metric	Hailfinder (DSL) CL metric
Grow-Shrink	$\alpha = 0.05$	229	327
	$\alpha = 0.1$	210	326
Hill Climbing		50	54
Max-Min Hill Climbing	$\alpha = 0.05$	194	237
	$\alpha = 0.1$	194	258
LUMIN	$NMI = 0.02, NCMI = 0.5$ with link removal	192	196
	$NMI = 0.02, NCMI = 0.7$ with link removal	161	206
	$NMI = 0.02, NCMI = 0.9$ with link removal	186	197
	$NMI = 0.01, NCMI = 0.9$ with link removal	209	217

Table 6.7: Points Lost by Learners from Hailfinder Network Data (smaller CL values are better)

learners for this dataset. It is clear that the DSL dataset is harder to learn from than that generated by the Bnlearn R package. The LUMIN learner seems to be comparable in performance to Grow-Shrink and Max-Min Hill Climbing learners. A particular problem with this dataset for LUMIN is the number of single parent children of the Scenario variable. This leads to a lot of very closely related variables which cannot be distinguished using MI alone. The graphs shown using some

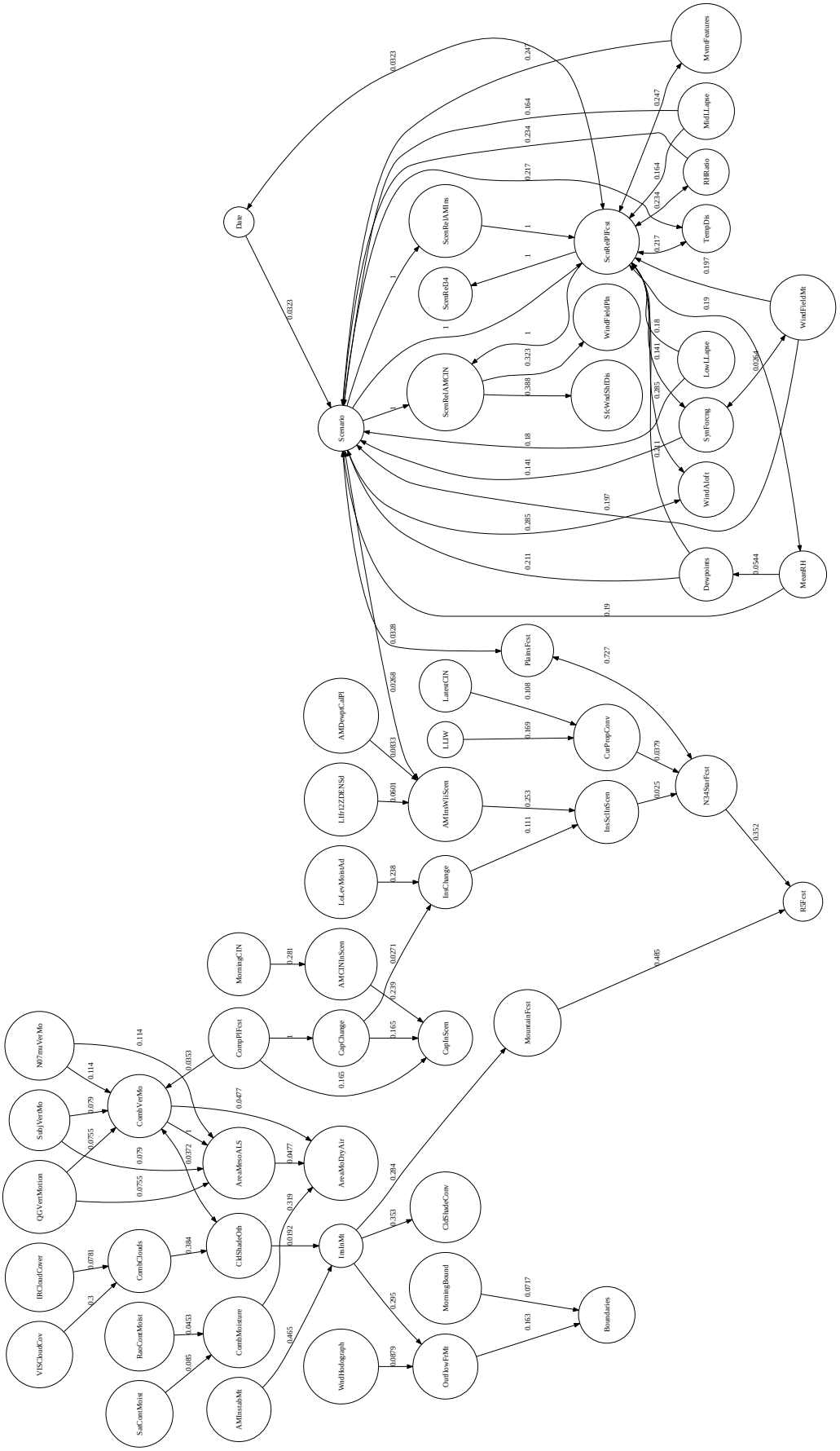


Figure 6.25: Hailfinder Network Learnt by the LUMIN Learner with $NMI = 0.01$, $NCMI = 0.9$ with Heuristic Link Removal

domain information demonstrate that the situation can be significantly improved using additional information although when the NMI between two variables approaches 1.0 their relationships will inevitably be confused if not explicitly specified or restricted.

Figures 6.26 and 6.27 show how the performance of the LUMIN learner varies over the two

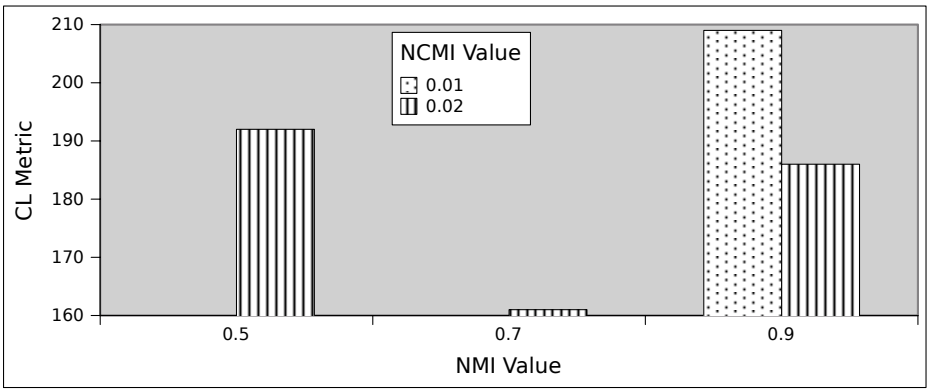


Figure 6.26: Graph of LUMIN's Performance for Hailfinder using the Bnlearn Dataset (smaller CL values are better)

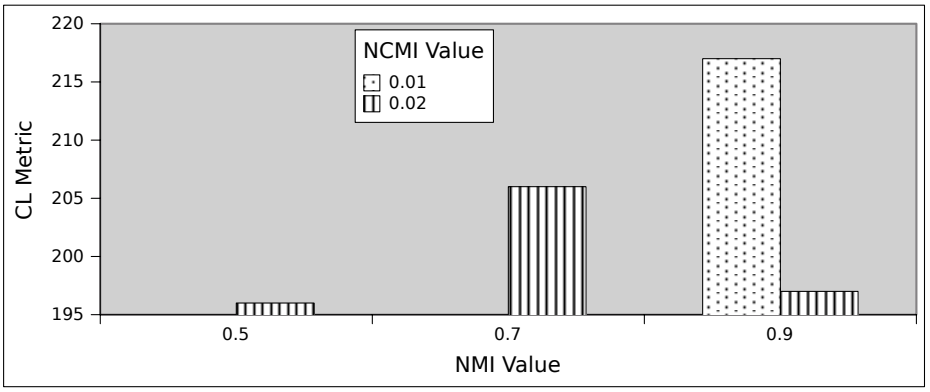


Figure 6.27: Graph of LUMIN's Performance for Hailfinder using the DSL Dataset (smaller CL values are better)

data sets with varying values of NMI and NCMI. The graphs have the lost CL metric points as the y-axis, smaller values are better, the NMI value used as the x-axis, and the NCMI value represented as shading within the columns. It is interesting to note that while an NMI value of around 0.7 appears to be optimum for the Bnlearn dataset, it seems to be a poor choice for the DSL dataset. Having performed only a relatively small number of tests its unclear what this indicates as while and NMI value of 0.7 was good the the Bnlearn dataset is was poor for the DSL dataset where NMI values of both 0.5 and 0.9 were superior to it. Figures 6.28 and 6.29 show a graphical representation of the performance of all the learners on the Bnlearn and DSL

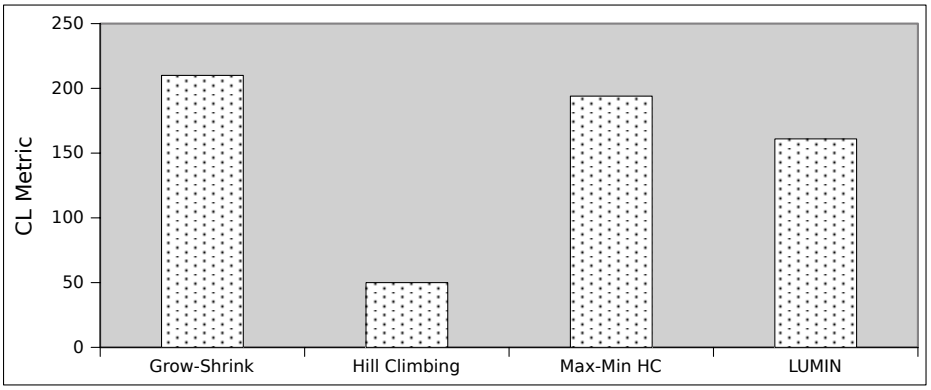


Figure 6.28: Graph of the Performance of all Learners for Hailfinder using the Bnlearn Dataset (smaller CL values are better)

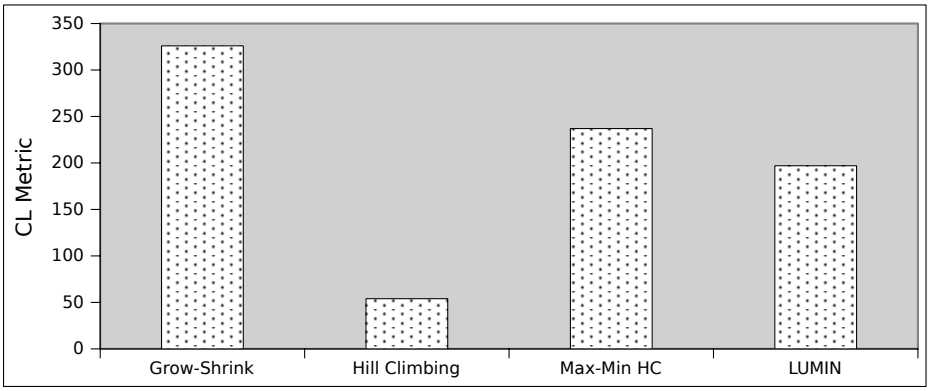


Figure 6.29: Graph of the Performance of all Learners for Hailfinder using the DSL Dataset (smaller CL values are better)

datasets respectively. It can be seen that despite the different optimal values for NMI on the two datasets it appears that the performance of LUMIN was in line with the of the other learners.

The Hailfinder network is not a typical network, in that it has many single parent children and a super parent, scenario, that is a parent to over a quarter of all the variables. The network has a number of distinct parts which are loosely connected, and both the grow-shrink and max-min hill climbing learners produced a number of separate networks rather than a single unified network. The Hailfinder network's variables have between 2 and 11 possible values. Therefore it is again possible that a single significance value for NMI may not be optimal. However, the LUMIN learner did produce results that were comparable to both the grow-shrink and max-min hill climbing learners, but the hill climbing learner was clearly the best performer on this network.

6.5 Insurance

The Insurance network introduced in [Binder et al 1997] is a network for car insurance risk estimation. It is a moderately complex network and is often used as a test network for learning systems. In the original paper 12 of the variables were hidden as the focus of the paper was in dealing with hidden variables.

The Insurance network consists of 27 variables with 52 relationships between them. The network is shown in figure 6.30. The variables involved are outlined in table 6.8 below. Similar

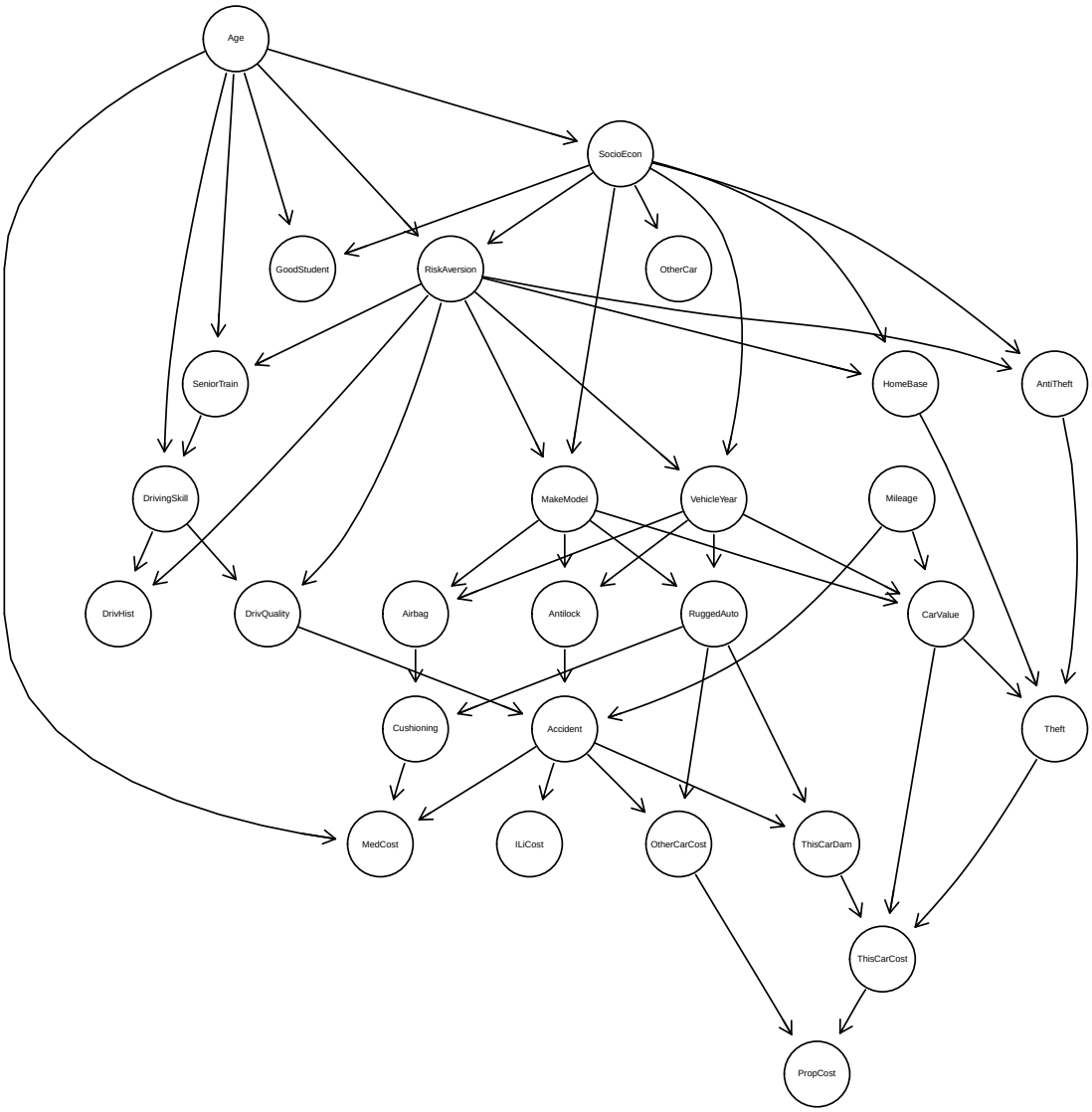


Figure 6.30: The Insurance Network

to the ALARM and Hailfinder networks the data items are categorical, that is, the variables can only take on a small number of fixed values as indicated above.

Table 6.8: Insurance Network Variables

Variable	Description	Possible Values
GoodStudent	good student	False and True
Age	age	Adolescent, Adult and Senior
SocioEcon	socio-economic status	Prole, Middle, UpperMiddle and Wealthy
RiskAversion	risk aversion	Psychopath, Adventurous, Normal and Cautious
VehicleYear	vehicle age	Current and Older
ThisCarDam	damage to this car	None, Mild, Moderate and Severe
RuggedAuto	ruggedness of the car	EggShell, Football and Tank
Accident	severity of the accident	None, Mild, Moderate and Severe
MakeModel	car's model	SportsCar, Economy, FamilySedan, Luxury and SuperLuxury
DrivQuality	driving quality	Poor, Normal and Excellent
Mileage	mileage	FiveThou, TwentyThou, FiftyThou and Domino
Antilock	ABS	False and True
DrivingSkill	driving skill	SubStandard, Normal and Expert
SeniorTrain	senior training	False and True
ThisCarCost	costs for the insured car	Thousand, TenThou, HundredThou and Million
Theft	theft	False and True
CarValue	value of the car	FiveThou, TenThou, TwentyThou, FiftyThou and Million
HomeBase	neighbourhood type	Secure, City, Suburb and Rural
AntiTheft	anti-theft system	False and True
PropCost	ratio of the cost for the two cars	Thousand, TenThou, HundredThou and Million
OtherCarCost	costs for the other car	Thousand, TenThou, HundredThou and Million
OtherCar	other cars involved in the accident	False and True
MedCost	cost of the medical treatment	Thousand, TenThou, HundredThou and Million
Cushioning	cushioning	Poor, Fair, Good and Excellent
Airbag	airbag	False and True
ILiCost	inspection cost	Thousand, TenThou, HundredThou and Million
DrivHist	driving history	Zero, One and Many

6.5.1 Results

We show the graphs learnt by the various different learners and discuss any issues which arose. As before we have two test datasets, one generated by the Bnlearn package and the other downloaded from DSL. The Bnlearn dataset has 20000 records and we combined four 5000 record datasets to make a 20000 record dataset from the DSL data[12].

6.5.2 The Bnlearn R Package - Bnlearn Data

As before we use a number of learners supplied by the Bnlearn R package. The insurance network obtained, using the optimised Grow-Shrink Markov Blanket learner with a target nominal type I error rate of 0.05 is shown in figure 6.31. The learner notes that the v-structures shown in table 6.9 have one or both links with contradictory direction assignments. The graph has 48 links

Airbag → AntiTheft ← RiskAversion	DrivingSkill → DrivHist ← MedCost
DrivingSkill → DrivHist ← ILiCost	DrivHist → MedCost ← OtherCarCost
DrivHist → ILiCost ← OtherCarCost	Age → RiskAversion ← AntiTheft
OtherCarCost → PropCost ← Theft	

Table 6.9: V-Structures with Contradictory Link Direction Assignments for the Bnlearn Insurance Dataset

between the nodes. With a target nominal type I error rate of 0.1 that graph shown in figure B.32 is learnt. At this error level the same v-structures cause problems with the exception of Age → RiskAversion ← AntiTheft, and the graph now has 58 links. The Hill Climbing learner produced the graph shown in figure 6.32, this contains 50 links and no isolated nodes. The Max-Min Hill Climbing learner produced the graph shown in figures 6.33 and B.33 both of which have 44 links and were generated using alpha values of 0.05 and 0.1 respectively.

6.5.3 The LUMIN Learner - Bnlearn Data

We start with a graph learnt by the LUMIN learner without using heuristic link removal. Figure B.24 shows the graph learnt with an NMI threshold of 0.03 and an NCMI threshold of 0.5. The graph has 126 links and no isolated nodes. Figure 6.34 shows a graph learnt with the same parameters, but allowing heuristic link removal. This graph has 35 links and no isolated nodes. Up to this point we have avoided using any domain information to aid in learning the networks partly as it makes comparison with the other learners more difficult, and partly because we don't

[12]The DSL data can be downloaded from `http://www.dsl-lab.org/supplements/mmhc_paper/ins_data.zip`.

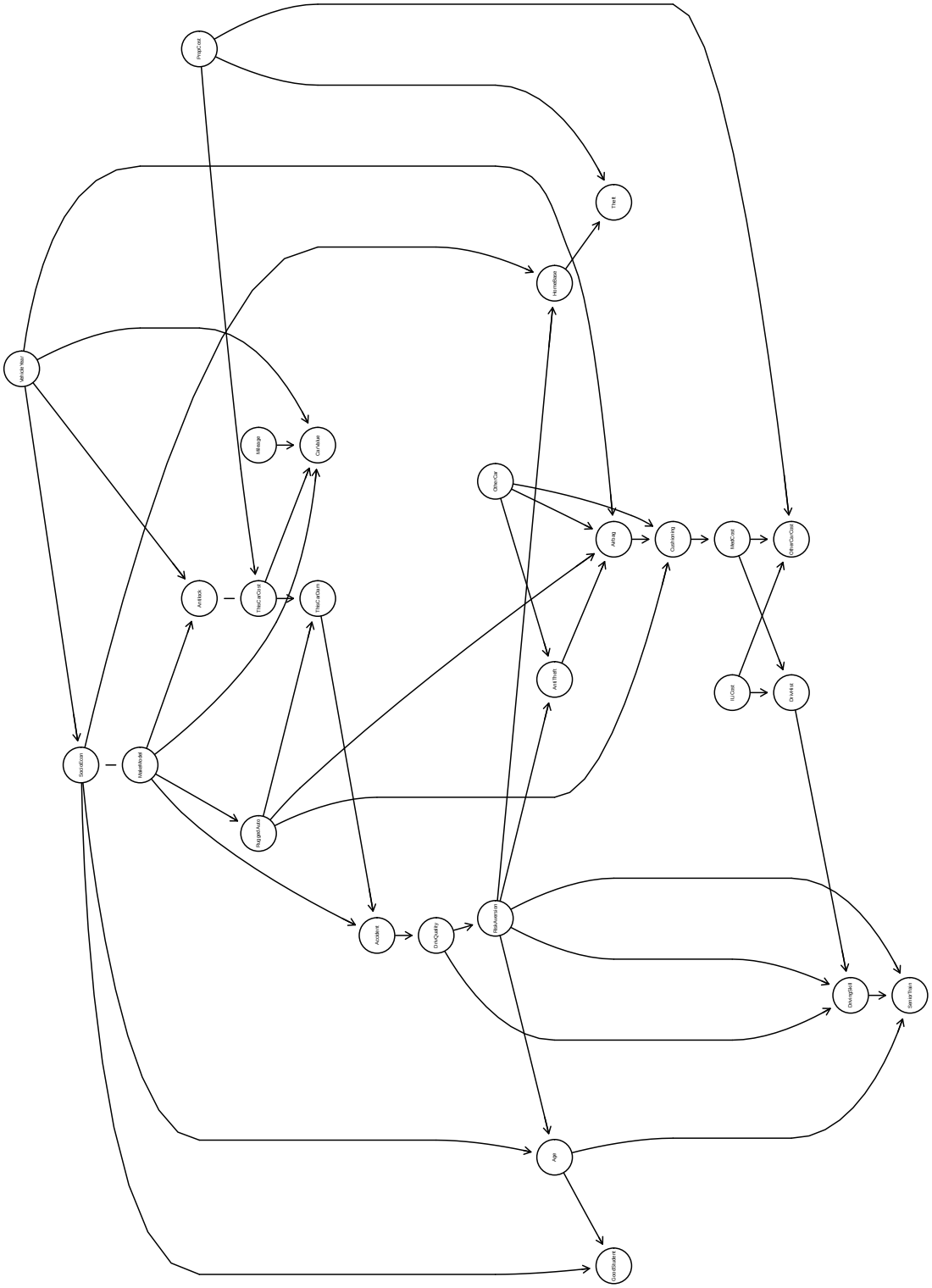


Figure 6.31: Insurance Network Learnt by the Grow-Shrink Learner with $\alpha = 0.05$

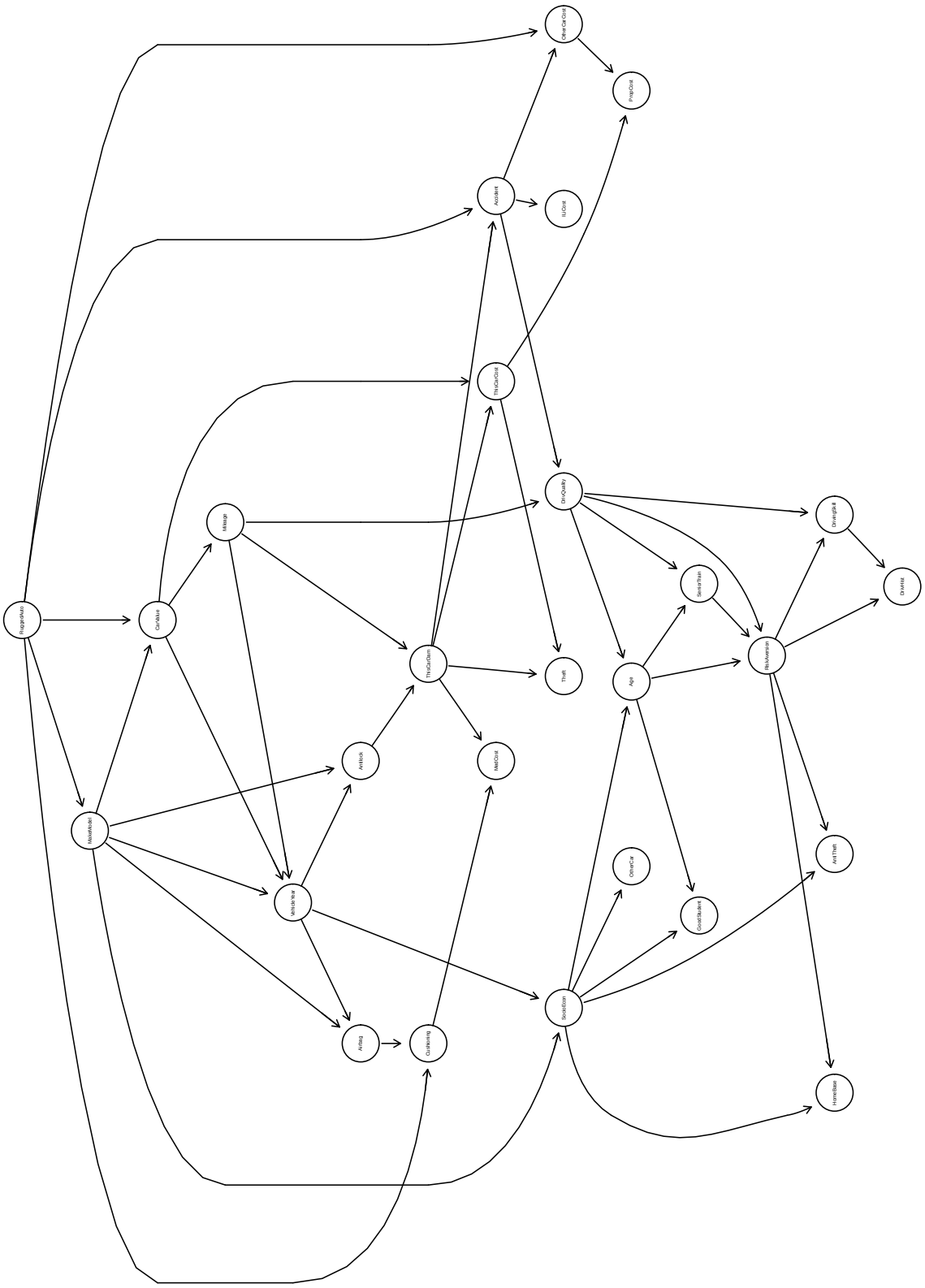


Figure 6.32: Insurance Network Learnt by the Hill Climbing Learner

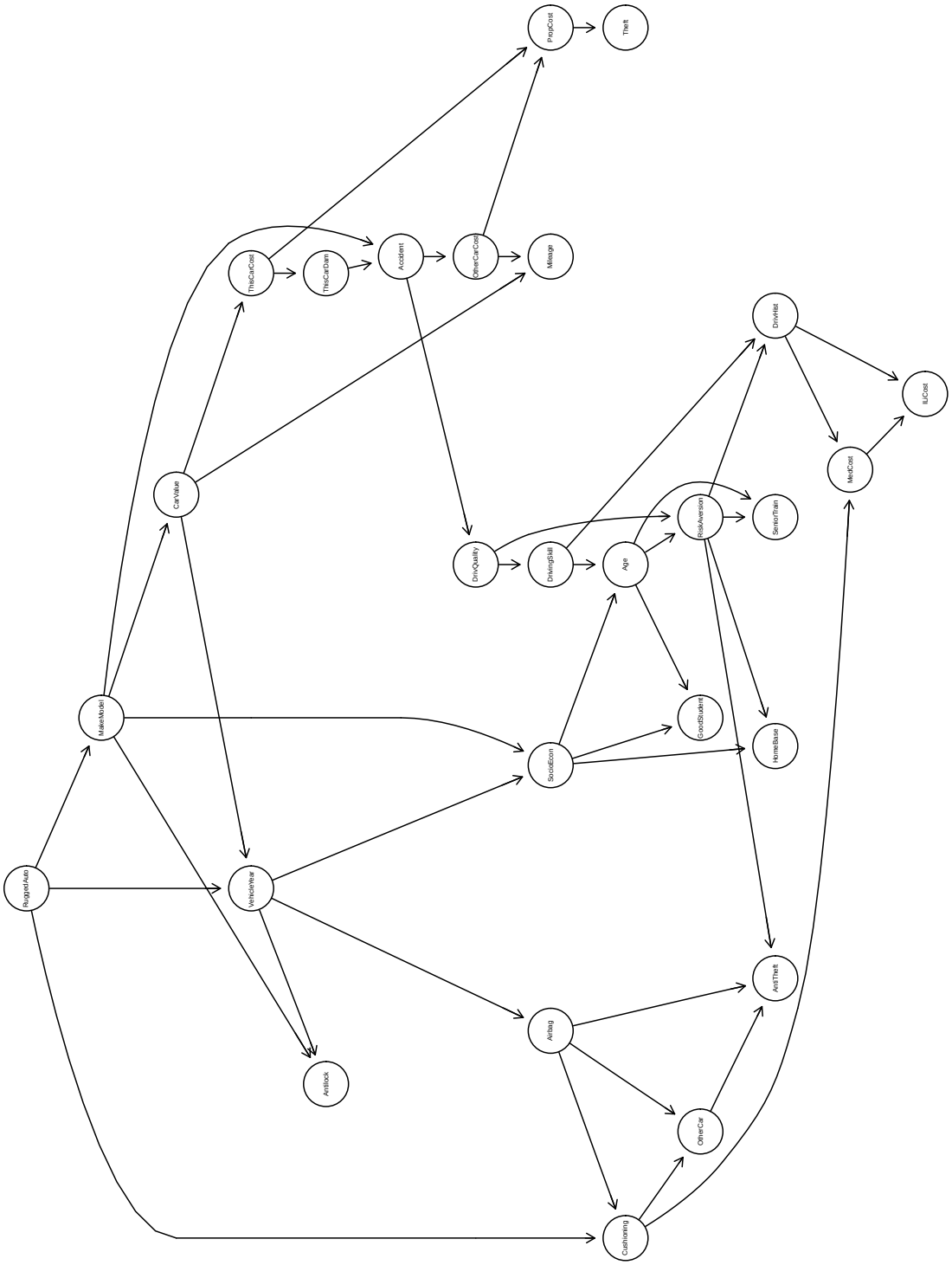


Figure 6.33: Insurance Network Learnt by the Max-Min Hill Climbing Learner with $\alpha = 0.05$

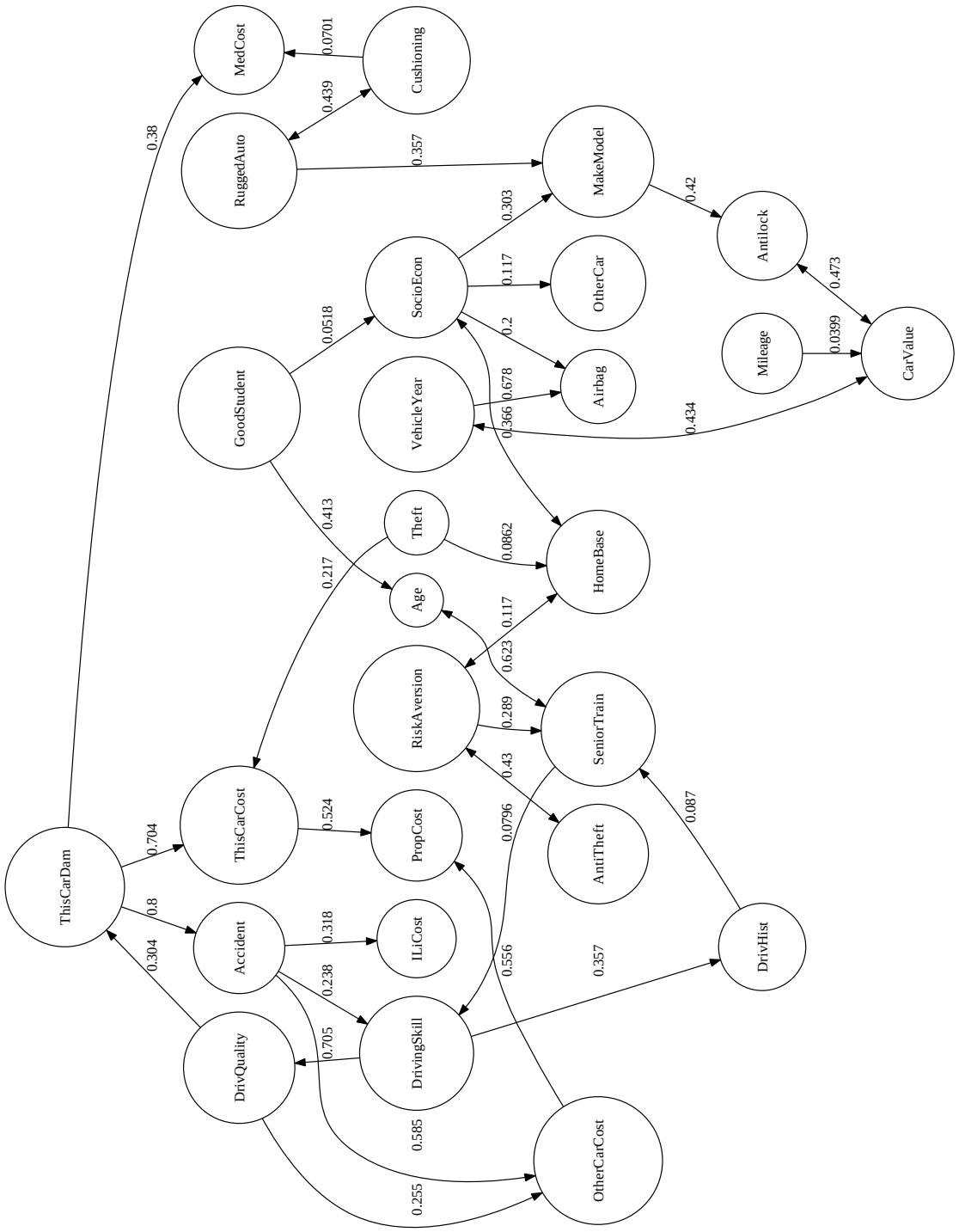


Figure 6.34: Insurance Network Learnt by the LUMIN Learner with $NMI = 0.03$ and $NCMI = 0.5$ with Heuristic Link Removal

really understand the variables in the networks well enough to make sensible predictions of their possible causal relationships. However, with the Insurance network the variables are relatively simple to understand and it is possible to make some reasonable statements about potential relationships between them. At this point we will to introduce the use of domain information to show how it helps with the learning process.

The variables Age and Mileage are exogenous and should not be effected by any of the other variables, similarly PropCost, MedCost, ILiCost and DrivHist should not be the cause of any of the other variables. We constructed domain information that split the variables into three groups the exogenous variables, the general variables and the child variables with rankings of 5, 10 and 15 respectively. The created domain, named External, had the rankings as shown in table 6.10.

Ranking 5	Ranking 10	Ranking 15
Age	GoodStudent	PropCost
Mileage	SocioEcon	MedCost
	RiskAversion	ILiCost
	VehicleYear	DrivHist
	ThisCarDam	
	RuggedAuto	
	Accident	
	MakeModel	
	DrivQuality	
	Antilock	
	DrivingSkill	
	SeniorTrain	
	ThisCarCost	
	Theft	
	CarValue	
	HomeBase	
	AntiTheft	
	OtherCarCost	
	OtherCar	
	Cushioning	
	Airbag	

Table 6.10: The External Domain for the Insurance Network

Next looking at the car related information it is clear that VehicleYear and MakeModel cannot be caused by any of RuggedAuto, Antilock, CarValue or Airbag. So, a separate domain was created, called CarPart, with the domain rankings shown in table 6.11.

Then looking at the accident and its causes and consequences we know that, if related, Accident must be the cause of ThisCarDam, OtherCarCost, MedCost and ILiCost. Similarly if any of RiskAversion, DrivQuality, DrivingSkill and SeniorTrain are related to Accident it must be as

Ranking 5	Ranking 10
VehicleYear	RuggedAuto
MakeModel	Antilock
	CarValue
	Airbag

Table 6.11: The CarPart Domain for the Insurance Network

causes. MedCost and ILiCost have already been identified as possible children to Accident in a previous domain so they can be left out of this one. So another domain, called Accident, was created with the rankings shown in table 6.12.

Ranking 5	Ranking 10	Ranking 15
RiskAversion	Accident	ThisCarDam
DrivQuality		OtherCarCost
DrivingSkill		
SeniorTrain		

Table 6.12: The Accident Domain for the Insurance Network

Looking at RiskAversion if causally related to VehicleYear, RuggedAuto, MakeModel, DrivQuality, Antilock, SeniorTrain, AntiTheft or Airbag it would be as a cause. Thus a new domain, called RiskAversion, was created with rankings shown in table 6.13.

Ranking 5	Ranking 10
RiskAversion	VehicleYear
	RuggedAuto
	MakeModel
	DrivQuality
	Antilock
	SeniorTrain
	AntiTheft
	Airbag

Table 6.13: The RiskAversion Domain for the Insurance Network

Lastly it is clear that AntiTheft can only be a cause of Theft and not visa versa so a domain was created with AntiTheft having a ranking of 5 and Theft having a ranking of 10. It may be possible to construct further domains with SocioEcon and HomeBase for example, but we are not as clear that there are definite obvious preexisting causal relationships for these and so left all other relations with their default rankings.

Figure 6.35 shows the graph learnt using the above domain information with an NMI threshold of 0.03 and an NCMI threshold of 0.5, the graph has no isolated nodes and 36 links. While this graph is an improvement over figure 6.34, see results in table 6.14, it is perhaps not as good

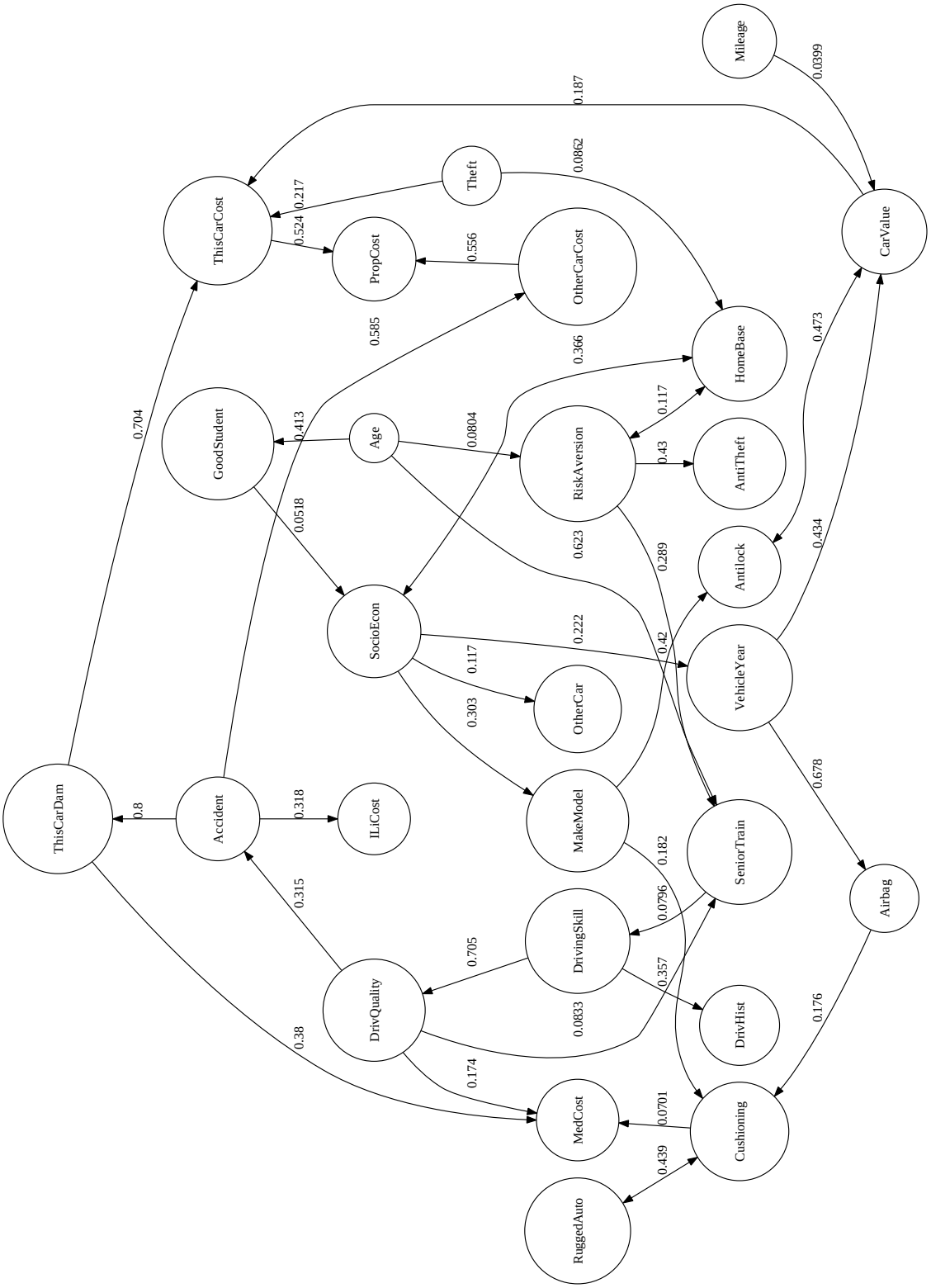


Figure 6.35: Insurance Network Learnt by the LUMIN Learner with $NMI = 0.03$ and $NCMI = 0.5$ with Heuristic Link Removal and Limited Domain Information

as it should be. The heuristic link removal incorrectly identifies a number of links as being due to a dominant ancestor, such as the links between DrivHist and RiskAversion and that between Airbag and MakeModel. Overall the use of the domain information improves the quality of the graphs learnt. Figures B.34, and B.35 show the graphs learnt by LUMIN using heuristic link removal and the above domain information with an NMI threshold of 0.03 and NCMI thresholds of 0.7 and 0.9 respectively. These graphs have no isolated nodes and they both have 34 links. Lastly as before figure B.36 shows the graph learnt by LUMIN with heuristic link removal, domain information and an NMI threshold of 0.01 with and NCMI threshold of 0.9. This graphs has no isolated nodes and 35 links.

6.5.4 The Bnlearn R Package - DSL Data

The network obtained, using the optimised Grow-Shrink Markov Blanket learner with a target nominal type I error rate of 0.05 is shown in figure 6.36. There were 9 v-structures which generated bi-directed links and 52 links in total. Figure B.37 shows the graph learnt by the optimised Grow-Shrink Markov Blanket learner with a target nominal type I error rate of 0.1, there were again 9 v-structures which generated bi-directed links, and 55 links in total. The Hill Climbing learner produced the graph show in figure 6.37 which has 50 links. The Max-Min Hill Climbing learner produced the graph shown in figure 6.38 with $\alpha = 0.05$. This graphs has 42 links. MMHC produced the graph show in figure B.38 with 42 links when the α value was 0.1. In this instance changing the value of α from 0.05 to 0.1 had no effect on the graph produced by the Max-Min Hill Climbing learner.

6.5.5 The LUMIN Learner - DSL Data

We start with the graph produced by the LUMIN learner with an NMI threshold of 0.03 an NCMI threshold of 0.5 and no heuristic link removal, this is shown in figure B.39. This network has 128 links and no isolated nodes. As is usual with no heuristic link removal there are an excessive number of incorrect links. Figure 6.39 shows the network learnt with the same thresholds, but using heuristic link removal. The network has no isolated nodes and 35 links. Using the same threshold values and including the domain information outlined above, LUMIN produces the graph shown in figure 6.40. This graph has no isolated nodes and 34 links. Figures B.40 and B.42 shows the graphs learnt by LUMIN with an NMI threshold of 0.03 and NCMI thresholds of 0.7 and 0.9 respectively. The graphs both have 35 links and no isolated nodes. Lastly figure

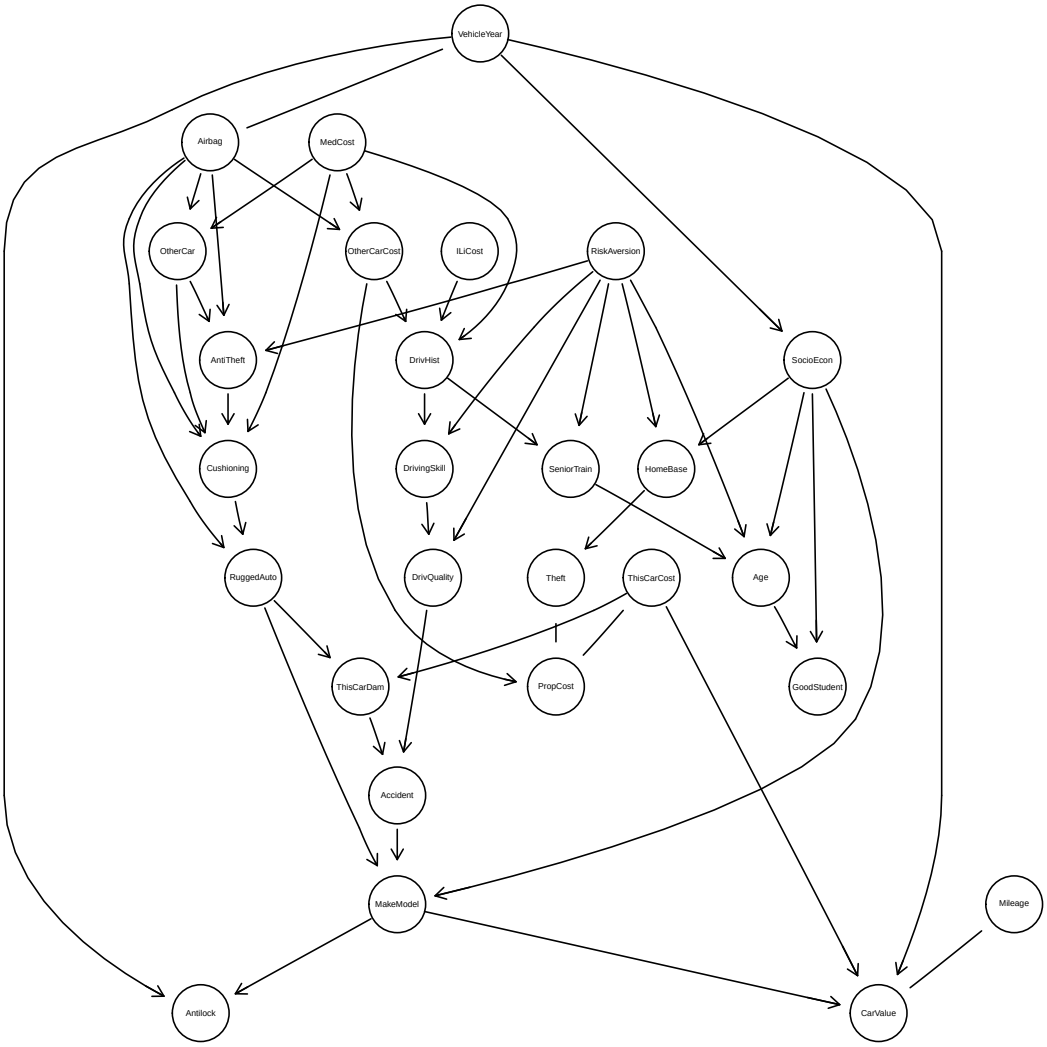


Figure 6.36: Insurance Network Learnt by the Grow-Shrink Learner with $\alpha = 0.05$

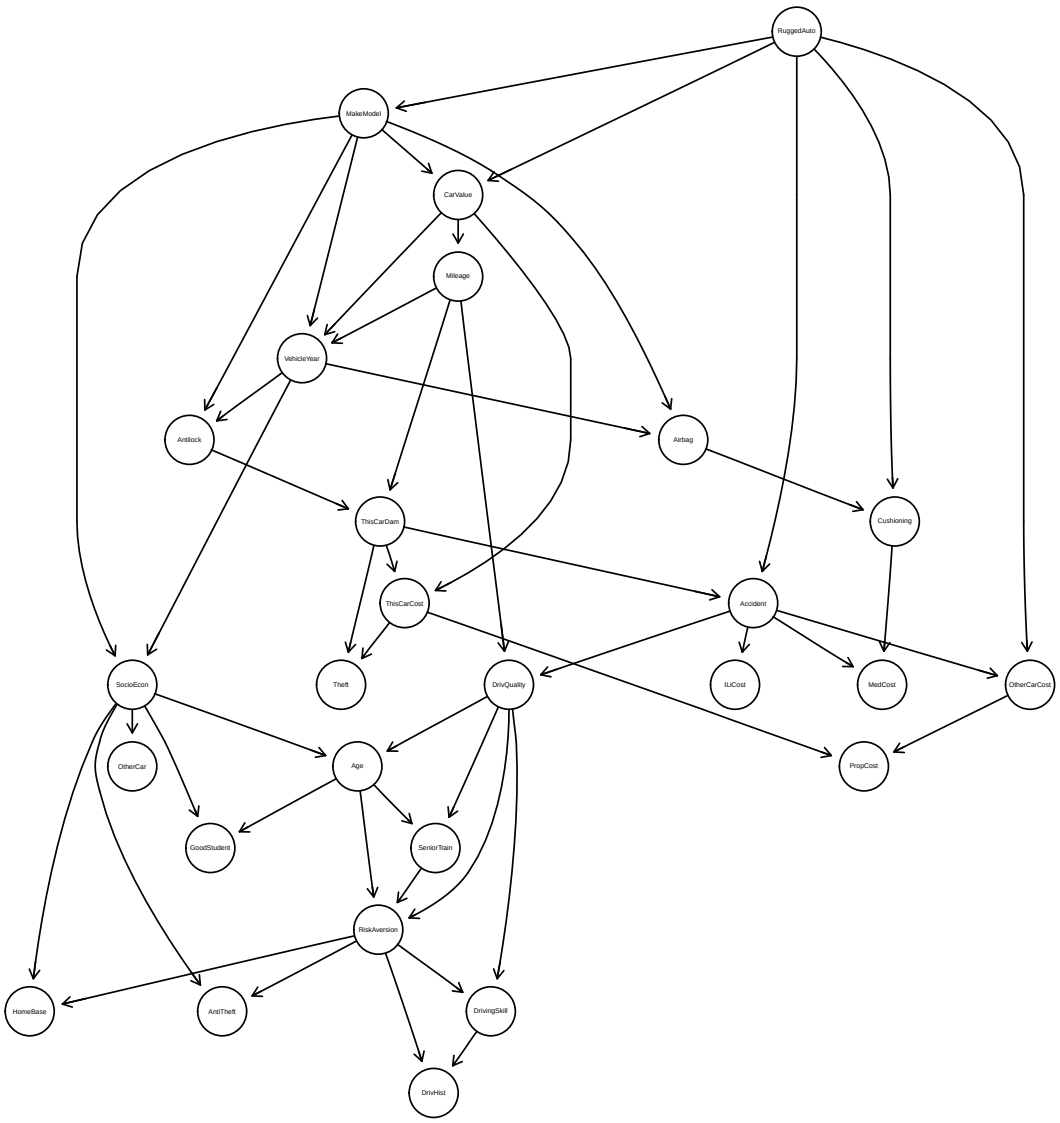


Figure 6.37: Insurance Network Learnt by the Hill Climbing Learner

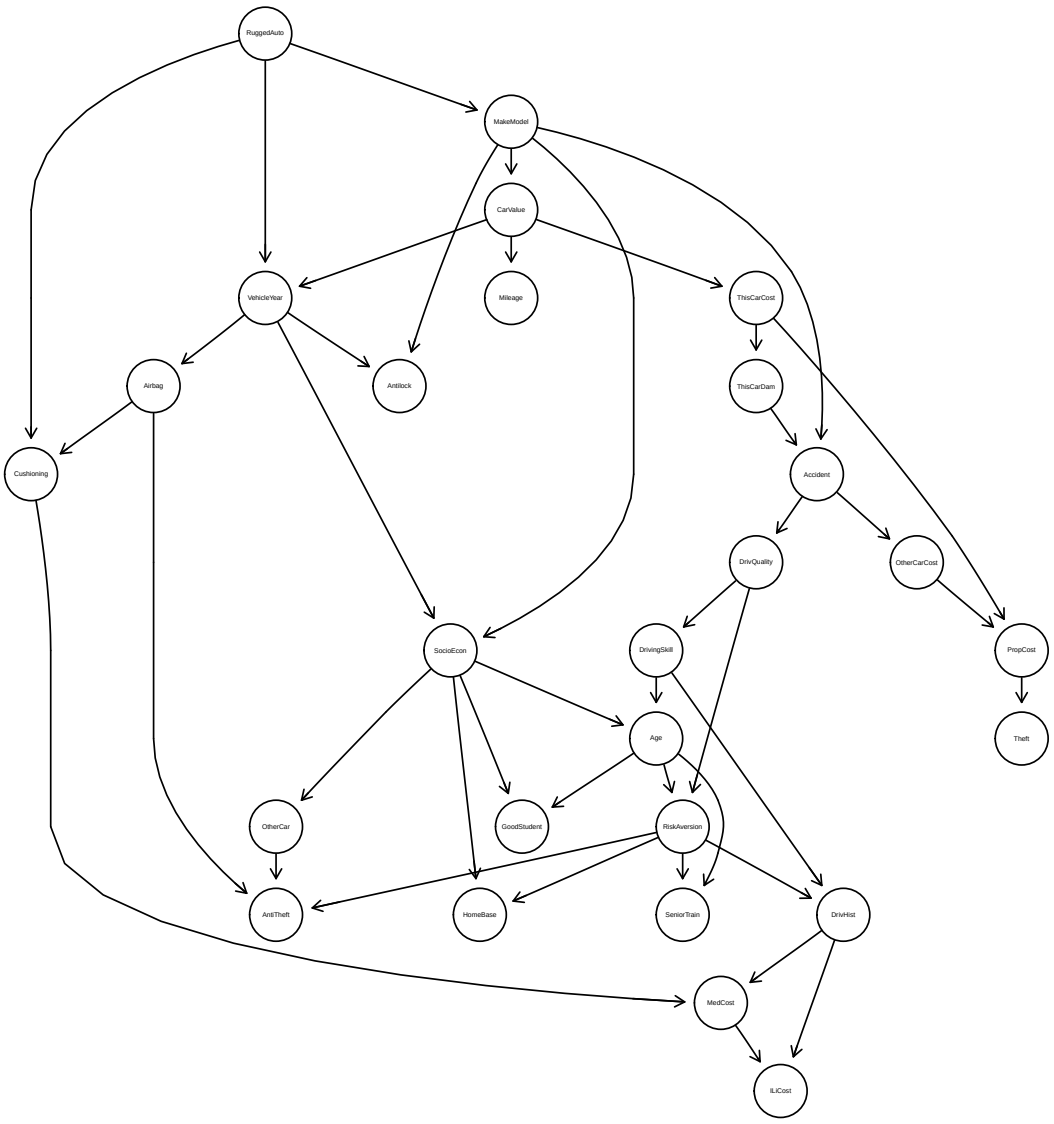


Figure 6.38: Insurance Network Learnt by the Max-Min Hill Climbing Learner with $\alpha = 0.05$

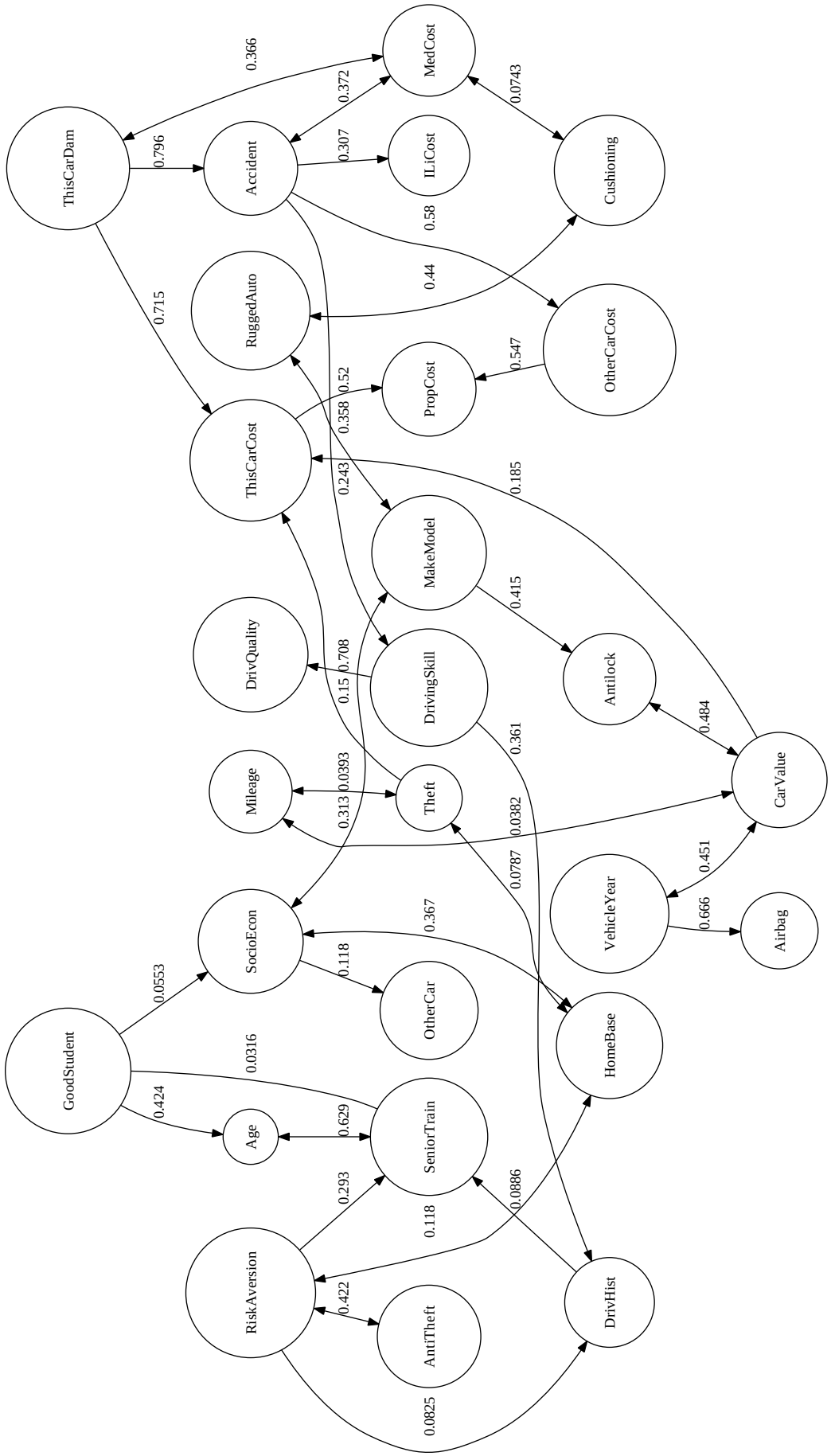


Figure 6.39: Insurance Network Learnt by the LUMIN Learner with $NMI = 0.03$ and $NCMI = 0.5$ with Heuristic Link Removal

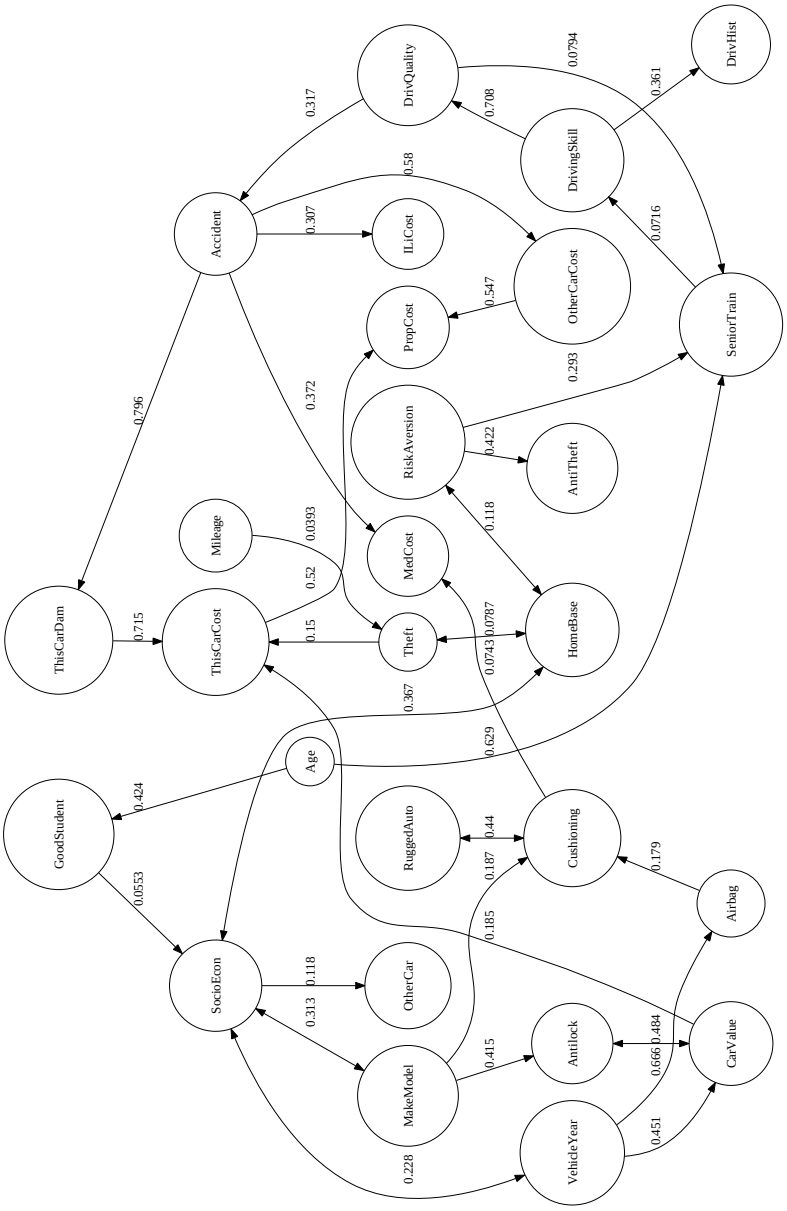


Figure 6.40: Insurance Network Learnt by the LUMIN Learner with $NMI = 0.03$ and $NCMI = 0.5$ with Heuristic Link Removal and Domain Information

B.41 which shows the graph learnt with in NMI threshold of 0.01 and an NCMI threshold of 0.9. The graph has no isolated nodes and 32 links.

6.5.6 Summary of Results for the Insurance Network

We have analysed the graphs produced by each of the learners, the result is shown in table 6.14.

		Insurance(Bnlearn)		Insurance(DSL)	
		CL Metric	Hamming Distance	CL Metric	Hamming Distance
Grow-Shrink	$\alpha = 0.05$	180	90	188	96
	$\alpha = 0.1$	170	–	206	–
Hill Climbing		138	70	133	72
Max-Min Hill Climbing	$\alpha = 0.05$	156	82	139	74
	$\alpha = 0.1$	156	–	139	–
LUMIN	$NMI = 0.03, NCMI = 0.5$ with link removal	164	77	155	72
	$NMI = 0.03, NCMI = 0.5$ with link removal and domain info	132	–	133	–
	$NMI = 0.03, NCMI = 0.7$ with link removal and domain info	132	–	132	–
	$NMI = 0.03, NCMI = 0.9$ with link removal and domain info	119	56	125	55
	$NMI = 0.01, NCMI = 0.9$ with link removal and domain info	156	67	168	78

Table 6.14: Points Lost by Learners from the Insurance Network (smaller CL values are better)

The table shows again that without domain information LUMIN is comparable to the other learners, and that its performance improves when supplied with additional domain information. An alternative view can help to clarify the relative performance of the learners. Figure 6.41 shows

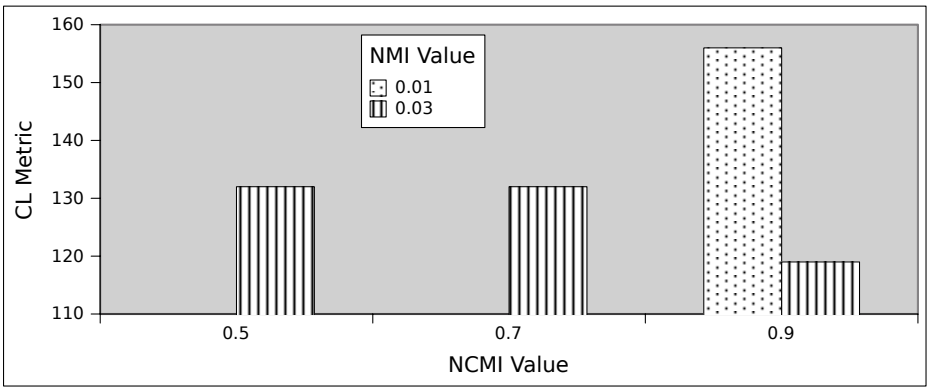


Figure 6.41: Graph of LUMIN Performance for Insurance using the Bnlearn Dataset (smaller CL values are better)

how the performance of the LUMIN learner on the Insurance network with data from the Bn-

learn package varies with differing values for NMI and NCMI. Similarly figure 6.42 shows the

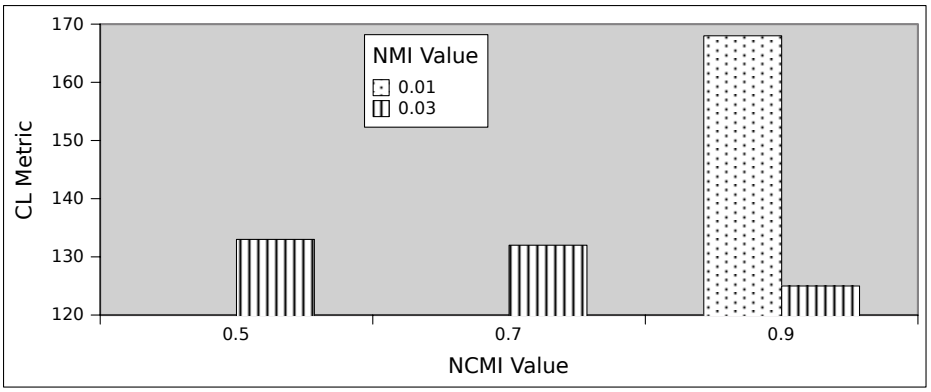


Figure 6.42: Graph of LUMIN Performance for Insurance using the DSL Dataset (smaller CL values are better)

performance of the LUMIN learner on the same network with the DSL data. These two graphs have the lost CL metric points as the y-axis, smaller values are better, the NMI value used as the x-axis, and the NCMI value represented as shading within the columns. This shows that for both datasets the higher NMI value of 0.9 gave the best performance suggesting that it was relatively easy to determine relationships between variables. The NCMI value of 0.03 was also best for both datasets. This small value for NCMI suggests that it was relatively hard to determine the direction of causality for some of the detected relationships. We can also compare the performance of the LUMIN learner with that of the other learners, in this instance we are using the graphs produced by the LUMIN learner when supplied with domain information. So, while LUMIN appears to perform very well on these datasets, we have given it an advantage although, as we reasoned the relationships supplied in the domain information rather than using what was already known, it is exactly the sort of usage for which LUMIN's domain and ranking system was designed. Figure 6.43 shows a comparison of the learner's performance with the Bnlearn dataset and figure 6.44 shows their performance with the DSL dataset.

The insurance network is another fairly standard network. Its variables range between 2 and 5 possible values. However, it does have one variable, RiskAversion, which is a parent of over 25% of the other variables, but, unlike the Hailfinder network super parent, this variable is not the sole parent of any of these variables. Since domain information was used in the case of the Insurance network, and improved its performance, it demonstrates that in cases where domain information is available it can be effectively used by the LUMIN learner. It is not unreasonable to assume that in the majority of cases when a causal learner will be useful there will be some

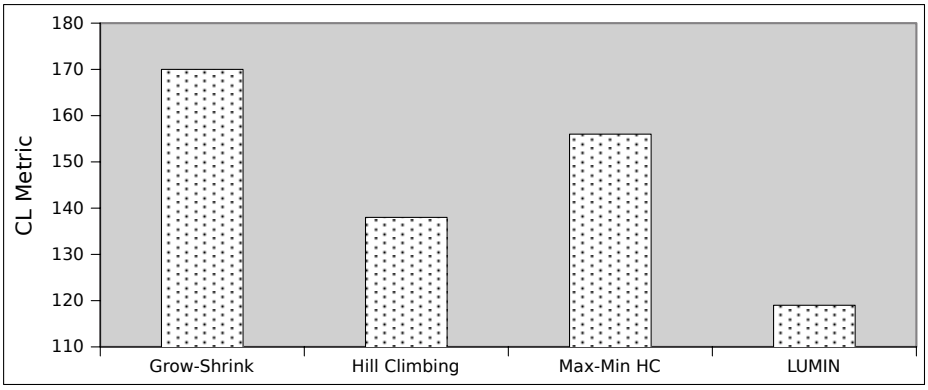


Figure 6.43: Graph of the Performance of all Learners for Insurance using the Bnlearn Dataset (smaller CL values are better)

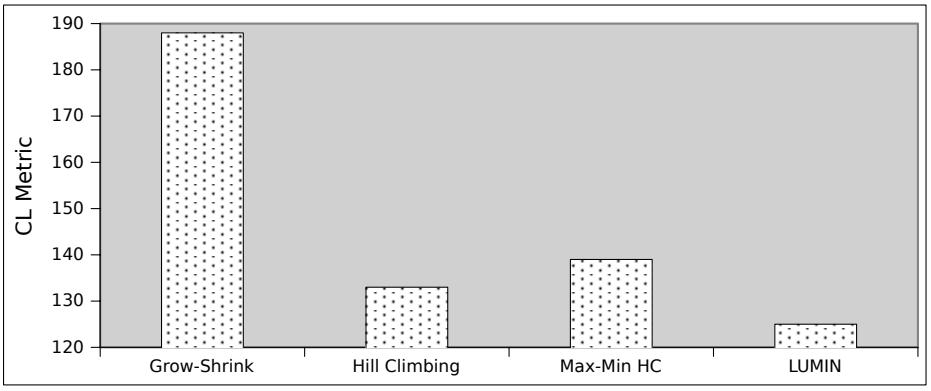


Figure 6.44: Graph of the Performance of all Learners for Insurance using the DSL Dataset (smaller CL values are better)

relevant information about some of the likely relationships in the data. A design aim was to allow easy use of any available domain information and this example goes some way towards showing that this is possible and effective with the LUMIN learner.

6.6 Non-Linear Relationships

A design aim for the LUMIN learner was to be able to discover non-linear relationships. Many learners assume linear relationships, see chapter 4, and although they may be able to discover non-linear relationships there is no guarantee that they will. The choice of mutual information as the basis of the LUMIN learner was partly driven by its known properties in being able to deal with non-linear relationships. To test this we created three simple networks whose relationships were all nonlinear. We started with four independent[13] real-valued variables, $A, B, C,$ and D and then created dependent variables using two or more of these independent variables.

6.6.1 A Polynomial Relationship

The first network used a polynomial as its relationship, the relationship used was:

$$N1 = A^4 - 1.2 * B^3 + 42 * C^2 \tag{6.1}$$

this is represented by the network shown in figure 6.45. A 20000 record dataset was created for

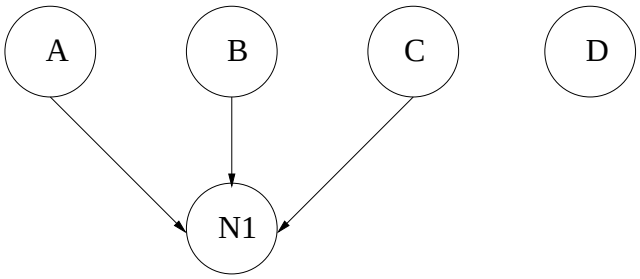


Figure 6.45: First Simple Non-Linear Network

this network using a Gaussian distribution for the sampling. Each of the learners when then tested with this dataset. The grow-shrink learner produced the graph shown in figure 6.46. The hill climbing and max-min hill climbing learners produced identical networks. All these networks miss the link between C and $N1$, but find the correct relationships for A, B and D with $N1$. The LUMIN learner produced the network shown in figure 6.47. In this case although LUMIN

[13] Independent within the limits of the pseudo-random number generator we were using.

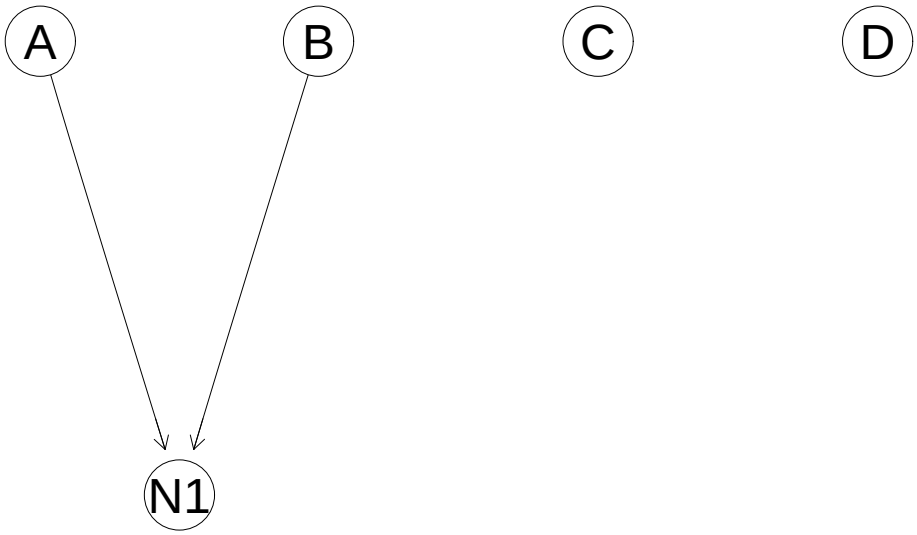


Figure 6.46: First Non-Linear Network Learnt by the Grow-Shrink Learner with $\alpha = 0.05$

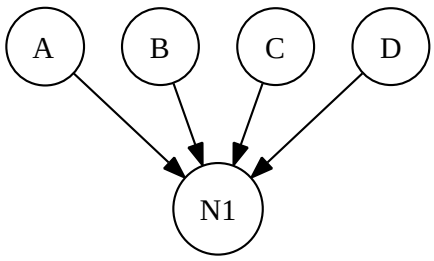


Figure 6.47: First Non-Linear Network Learnt by the LUMIN learner $NMI = 0.5, NCMI = 0.5$

correctly found the link between C and $N1$ we have an additional incorrect link between D and $N1$. Examining the output of LUMIN indicates that there is an order of magnitude more NMI between D and $N1$, 0.736 than there is between D and A, B, or C, 0.056. We cannot explain why this is, it may be due to a property of the random number generator, but it is the cause of the extraneous causal relationship.

6.6.2 A Trigonometric Relationship

The network for the second non-linear relationship is shown in figure 6.48. In this example we

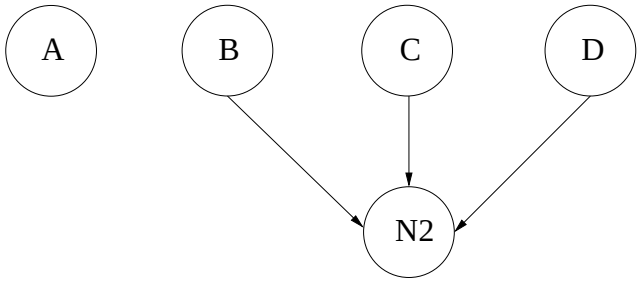


Figure 6.48: Second Simple Non-Linear Network

used trigonometric relationships, the relationships involved are:

$$N2 = 10 * (\cos(B) + \sin(C) + \cos(D)) \tag{6.2}$$

Once again we generated a 20000 record dataset with Gaussian distribution for the sampling. The grow-shrink learner produced the network shown in figure 6.49 with a target nominal type

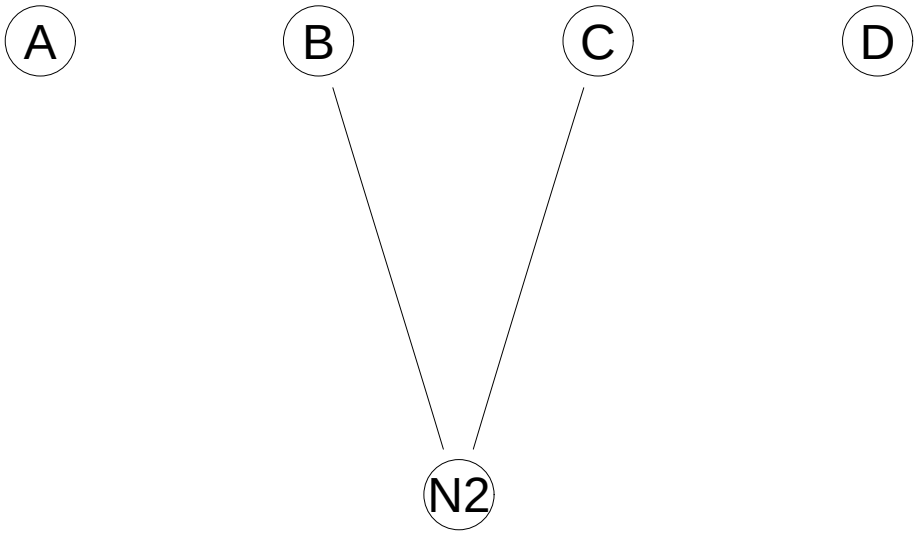


Figure 6.49: Second Non-Linear Network Learnt by the Grow-Shrink Learner with $\alpha = 0.1$

1 error rate of 0.1. The grow-shrink learner discovered there was some relationship between B, C and $N2$, but was unable to determine the direction of causality. The relationship between D and $N2$ was missed. The hill climbing and max-min hill climbing learners failed to detect any relationships between the variables. The LUMIN learner produced the graph shown in figure

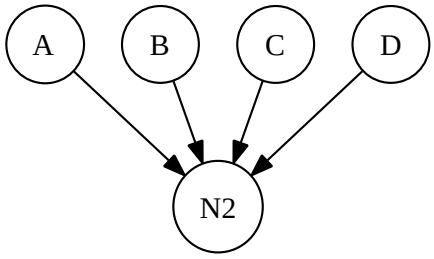


Figure 6.50: Second Non-Linear Network Learnt by the LUMIN learner $NMI = 0.2, NCMI = 0.5$

6.50 with an NMI value of 0.2 and an NCMI value of 0.5. While LUMIN correctly detected the relationships between B, C, D and $N2$ it incorrectly detected a relationship between A and $N2$. Once again this was due to an unexpectedly high value of NMI between A and $N2$, a value of 0.28, compare this value with the next highest NMI value for A of 0.056.

6.6.3 A Logarithmic Relationship

The network for the third non-linear relationship is shown in figure 6.51. This network used

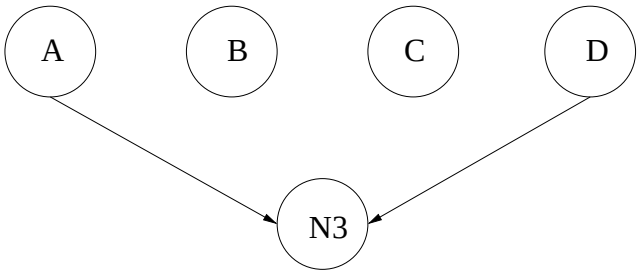


Figure 6.51: Third Simple Non-Linear Network

logarithmic relationships given by:

$$N3 = 10 * (\log(A) + \log(D)) \tag{6.3}$$

A dataset with 20000 entries was generated using a Gaussian distribution for sampling. All the learners successfully learnt the correct graph, LUMIN using an NMI value of 0.27 and an NCMI value of 0.5.

6.6.4 Combined Non-Linear Relationships

We combined the datasets to produce a single 20000 record dataset with all three relationships present to give the network shown in figure 6.52. The dashed lines between the variables $N1$, $N2$

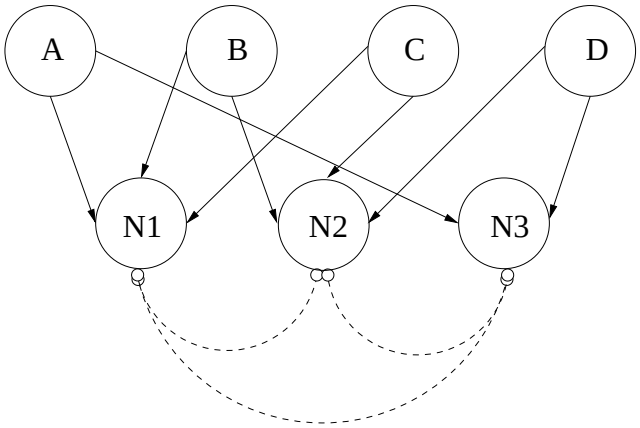


Figure 6.52: Combined Non-Linear Network

and $N3$ indicate that since all three are deterministically derived from some of the same variables they probably share some sort of non-causal relationship. The grow-shrink learner produced the graph shown in figure 6.53 with a target nominal type 1 error rate of 0.1. The hill climbing

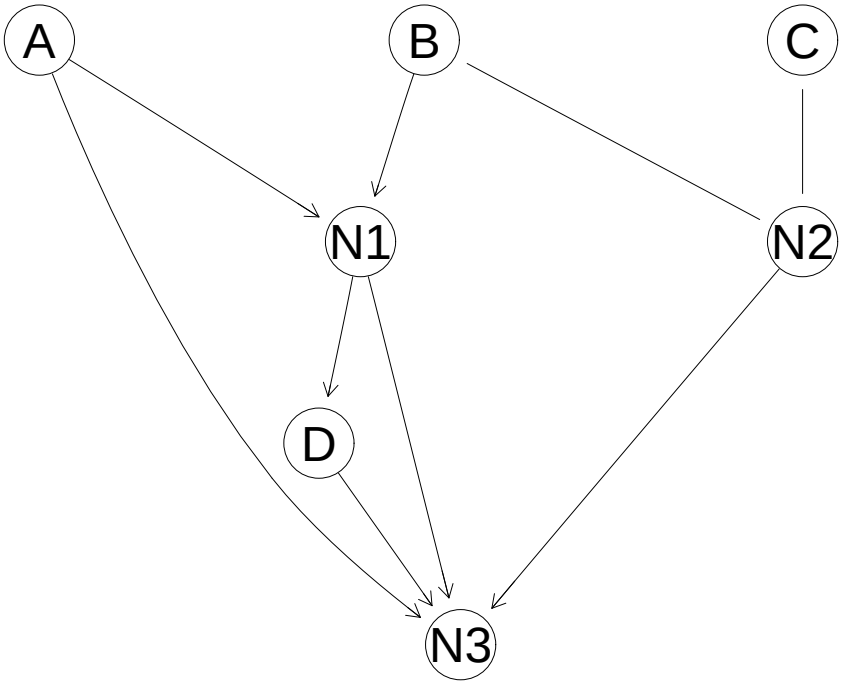


Figure 6.53: Combined Non-Linear Network Learnt by the Grow-Shrink Learner with $\alpha = 0.1$

learner produced the graph shown in figure 6.54. This was also the best graph that could be obtained with the max-min hill climbing learner with a the target nominal type 1 error rate of the conditional independence test of 0.1. The LUMIN learner produced the graph shown in figure

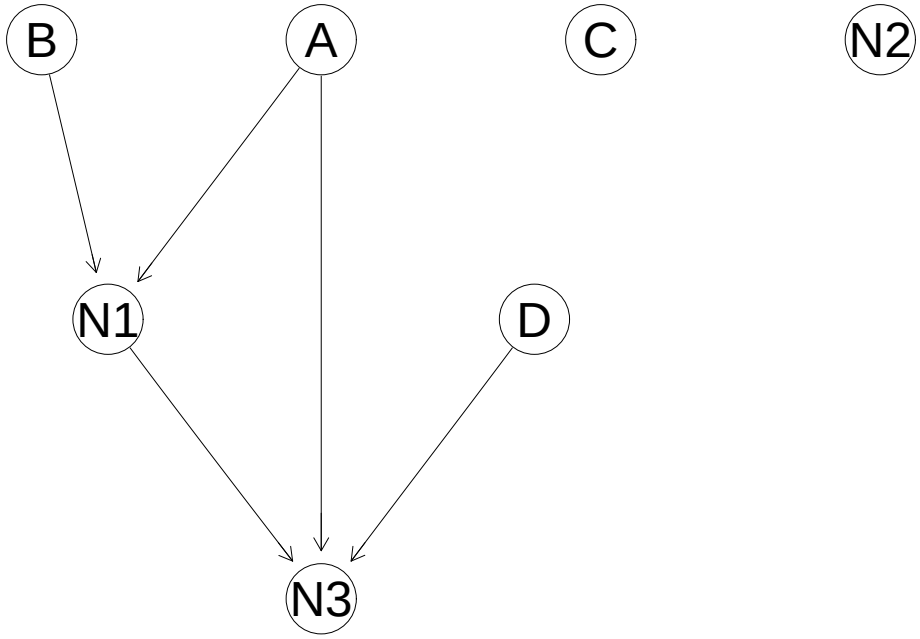


Figure 6.54: Combined Non-Linear Network Learnt by the Hill Climbing Learner

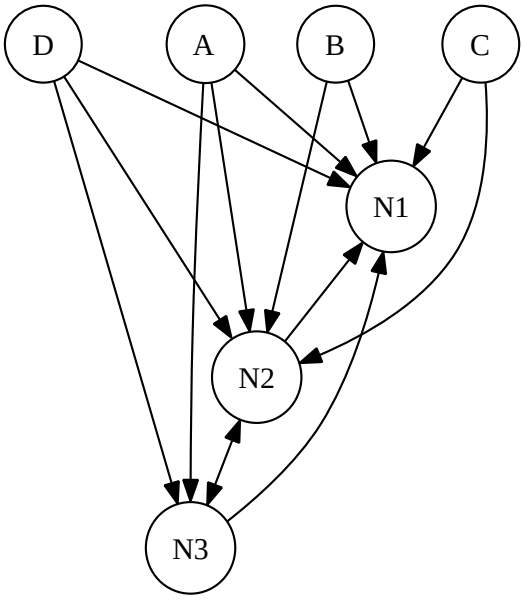


Figure 6.55: Combined Non-Linear Network Learnt by the LUMIN learner $NMI = 0.266, NCMI = 0.4$

6.55 with an NMI value of 0.266 and an NCMI value of 0.4.

A comparison of the performance of the learners can be made with the CL metric, as in the previous examples we will score them in terms of lost score rather than total score. Table 6.15

	First	Second	Third	Combined
Grow-Shrink	5	7	0	16
Hill Climbing	5	15	0	20
Max-Min Hill Climbing	5	15	0	20
LUMIN	4	4	0	8

Table 6.15: CL Metric Points Lost by Learners on Non-Linear Networks (smaller CL values are better)

shows the lost scores in terms of the CL metric for the three simple non-linear networks and the combined network. We do not include the relationships between the variables $N1$, $N2$ and $N3$ in the count for the combined network. An alternative view of these results is shown in figure 6.56.

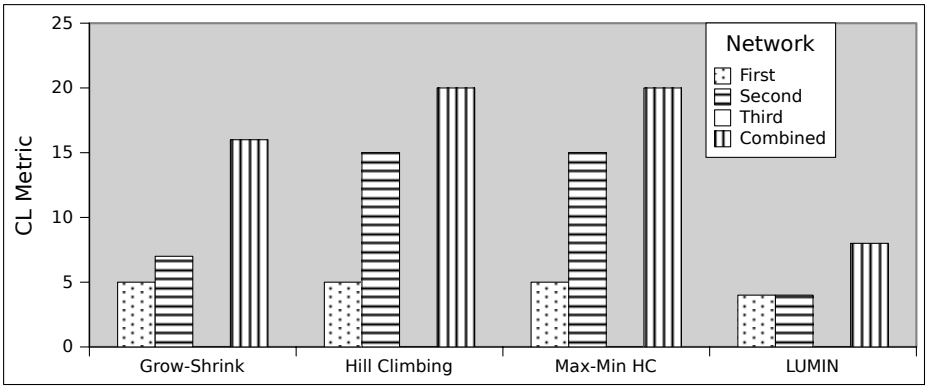


Figure 6.56: Graph of Performance of all Learners on Non-Linear Networks (smaller CL values are better)

So it appears that LUMIN is comparable or better at dealing with some simple non-linear relationships than the other learners tested. The most significant difference was with trigonometric relationships where both the hill climbing and max-min hill climbing learners failed to find any relationships. It is also interesting to note that combining the simple networks into a slightly more complex one did not appear to help or hinder most of the learners as their combined CL score was just the sum of the individual scores.

6.7 Causal Loops

Most causal learners make assumptions which make it impossible for them to discover causal loops, most commonly the Causal Markov assumption, see definition 20, section 2.9.3. The as-

sumptions behind the LUMIN learner should not preclude the discovery of causal loops, although there can still be difficulties in discovering them related to the how the data are presented to the learner, see sections 3.6 and 2.15. To examine how capable LUMIN is at discovering causal loops we created a number of simple networks which include a loop and examine how LUMIN and other learners cope with learning these networks.

6.7.1 First Causal Loop Test Network

Figure 6.57 shows the our first network test structure with a loop. As before variables A, B, C,

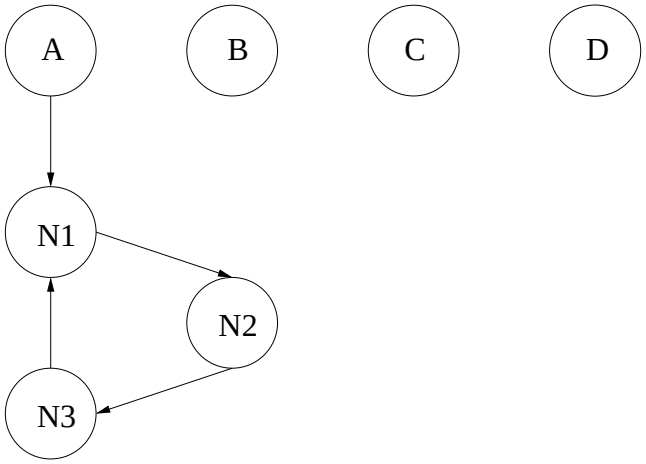


Figure 6.57: First Causal Loop Test Network

and D are independent and the variables labeled $N1 \ldots N3$ are derived from other independent or dependent variables. One issue here is that with only a single independent variable we would expect there to be relationships between A and both of $N2$ and $N3$. We produced a 20000 record dataset using a Gaussian distribution for sampling. The grow-shrink learner produced the graph shown in figure 6.58 with a target nominal type 1 error rate of 0.001. The hill climbing learner produced the network shown in figure 6.59. The max-min hill climbing learner produced the network shown in figure 6.60 with a the target nominal type 1 error rate of the conditional independence test of 0.001. The LUMIN learner produced the network shown in figure 6.61 with an NMI value of 0.08 and an NCMI value of 0.03.

6.7.2 Second Causal Loop Test Network

Figure 6.62 shows another network which includes a causal loop. Like the first causal loop test network the basic loop is still $N1 \rightarrow N2 \rightarrow N3 \rightarrow N1$, however, in this case both $N1$ and $N2$ have external drivers that should make the discovery of the loop easier. Once again we

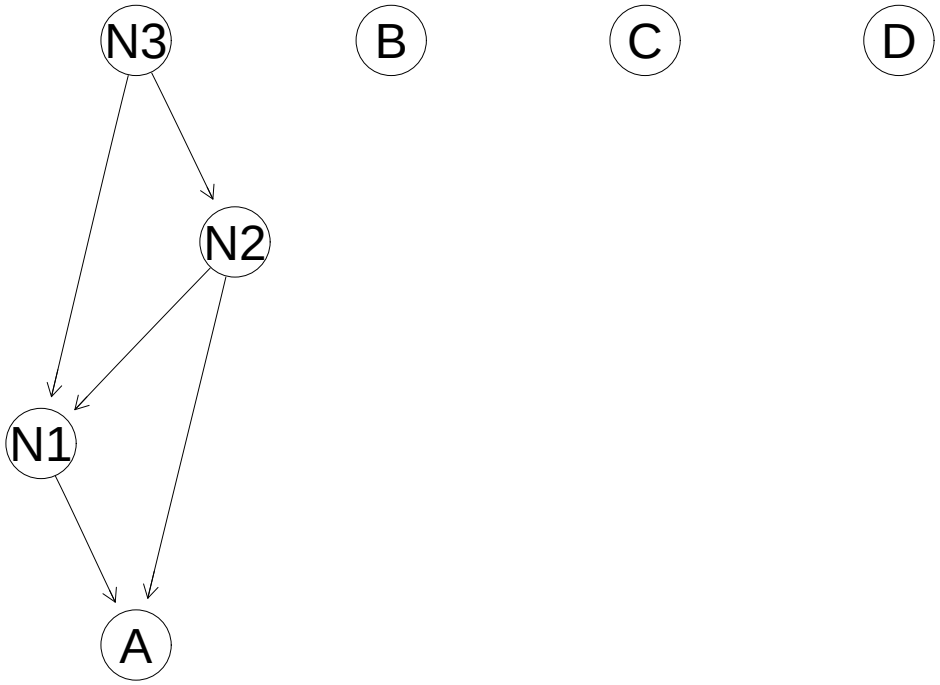


Figure 6.58: Network Learnt by Grow-Shrink Learner for First Causal Loop Network with $\alpha = 0.001$

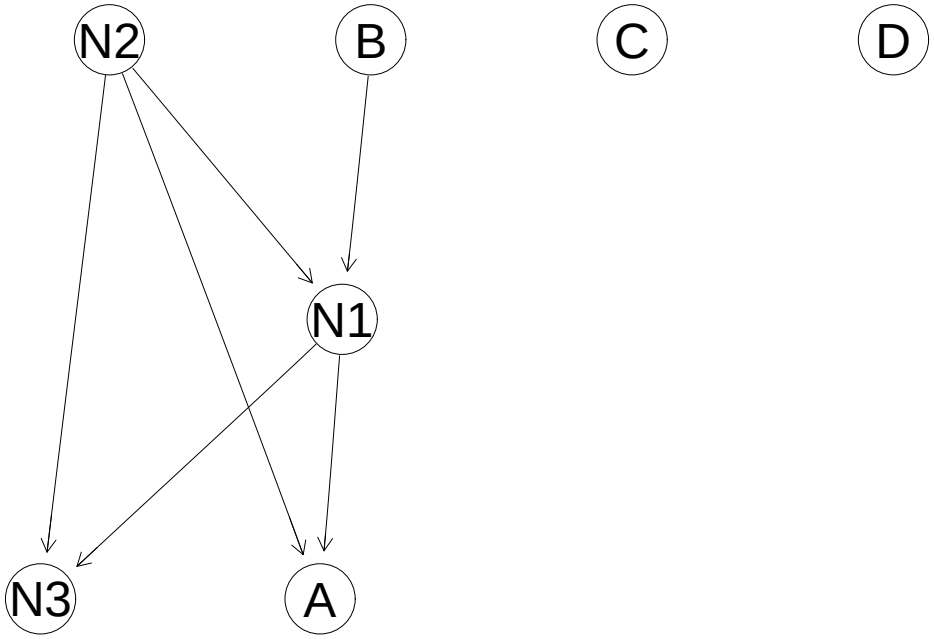


Figure 6.59: Network Learnt by Hill Climbing Learner for First Causal Loop Network

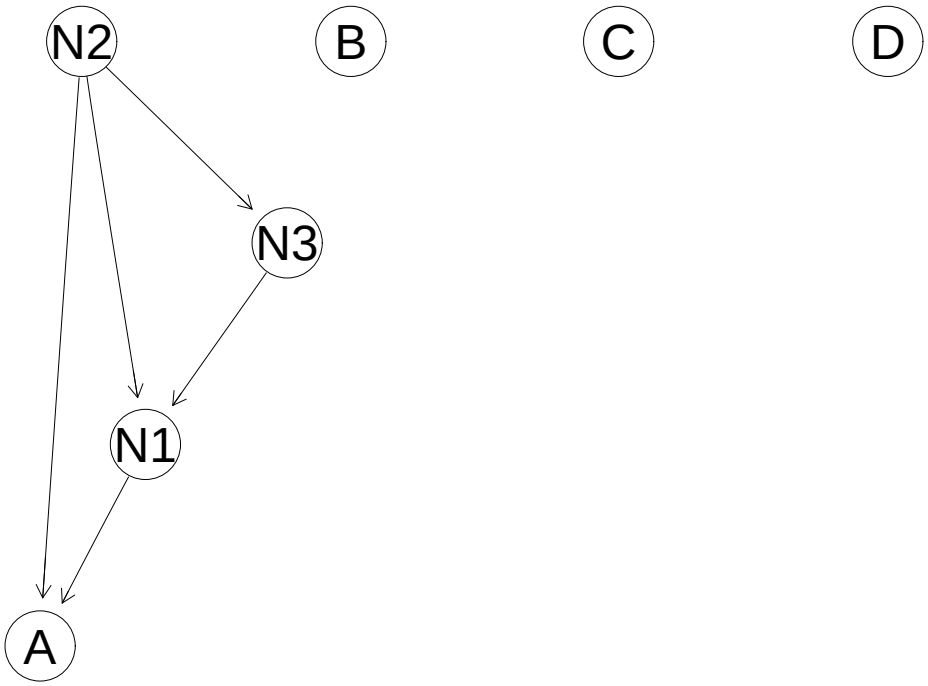


Figure 6.60: Network Learnt by Max-Min Hill Climbing Learner for First Causal Loop Network with $\alpha = 0.001$

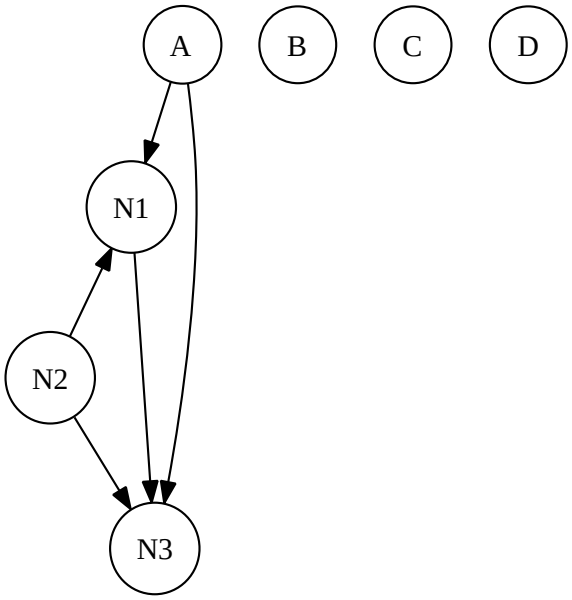


Figure 6.61: Network Learnt by the LUMIN Learner for First Causal Loop Network with $NMI = 0.08, NCMI = 0.3$

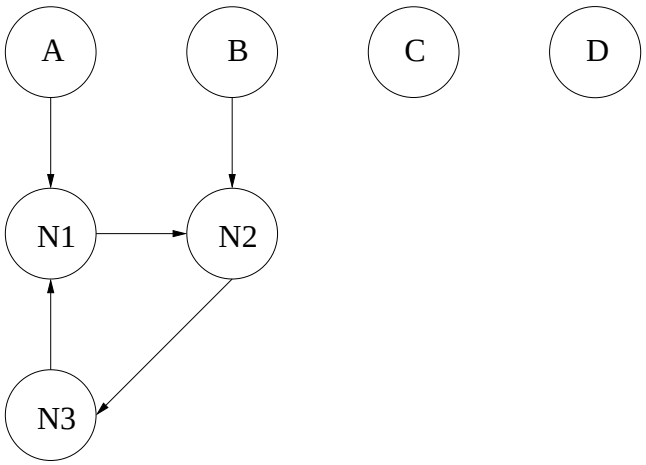


Figure 6.62: Second Causal Loop Test Network

produced a 20000 record dataset with a Gaussian distribution for sampling. The grow-shrink learner produced the graph shown in figure 6.63 and seemed fairly insensitive to the value of the

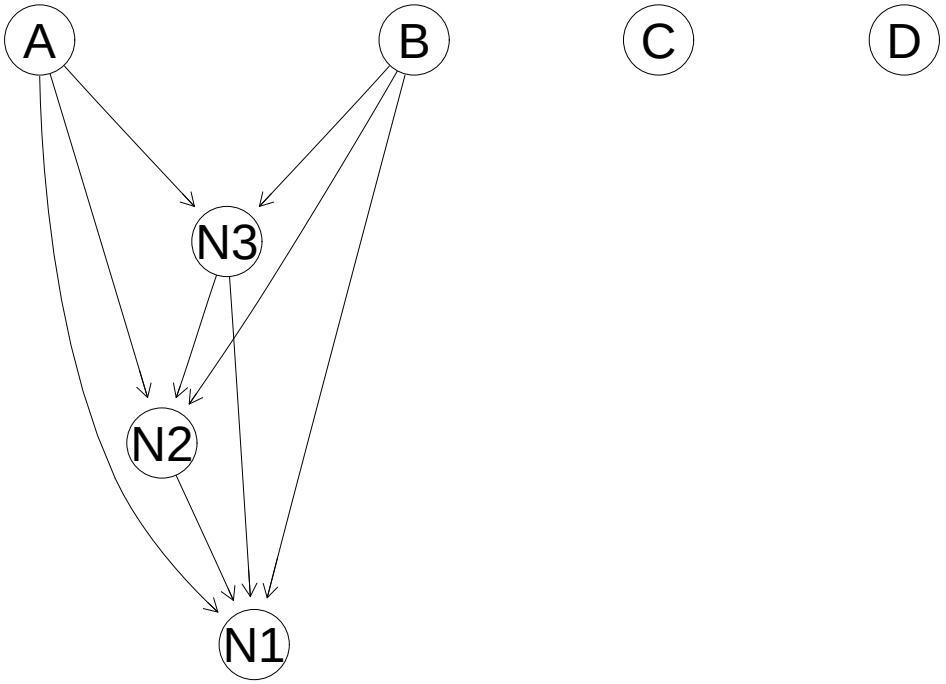


Figure 6.63: Network Learnt by Grow-Shrink Learner for Second Causal Loop Network

target nominal type 1 error rate. The hill climbing learner produced the network shown in figure 6.64. Like the grow-shrink learner the max-min hill climbing learner was fairly insensitive to the value of the target nominal type 1 error rate of the conditional independence test, and produced the graph shown in figure 6.65. In this instance LUMIN was able to show its greater flexibility. LUMIN produced the network shown in figure 6.66, and we can see that while it is not perfect, there is an indication of the presence of the loop in that the loop elements are joined by double

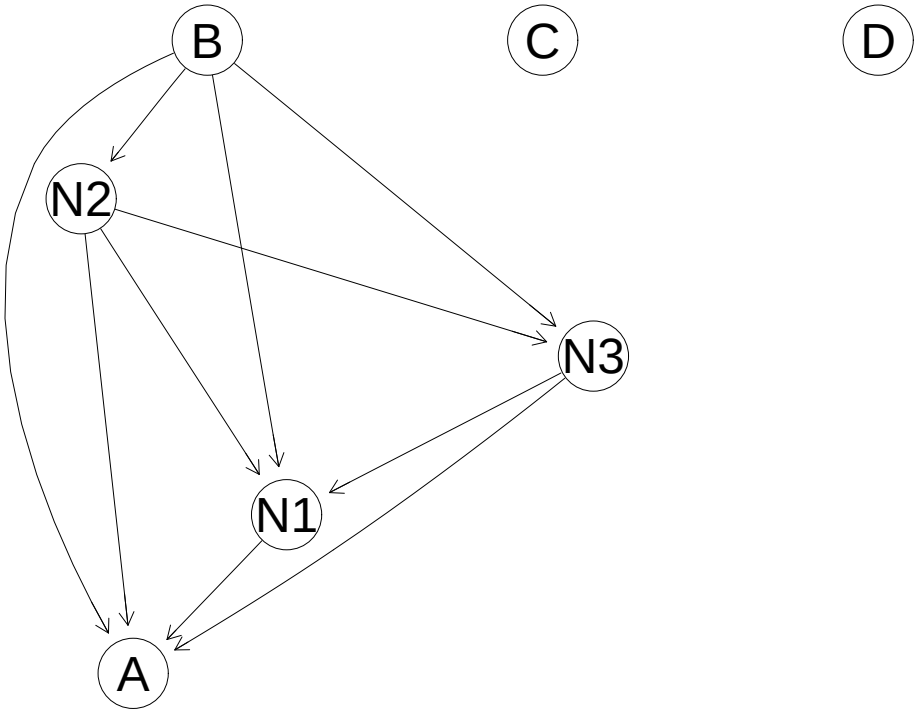


Figure 6.64: Network Learnt by Hill Climbing Learner for Second Causal Loop Network

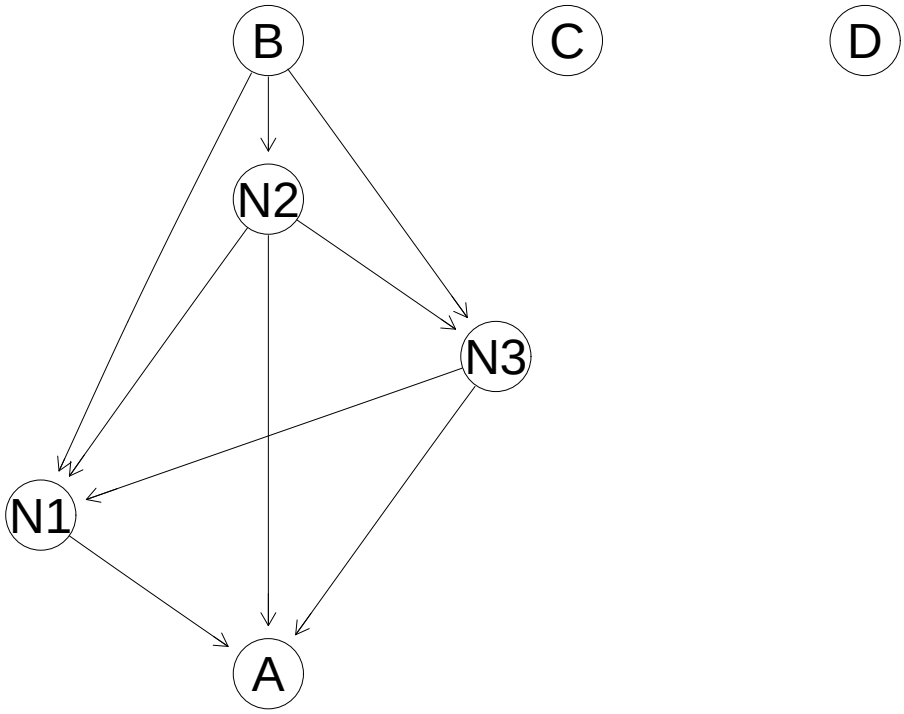


Figure 6.65: Network Learnt by Max-Min Hill Climbing Learner for Second Causal Loop Network

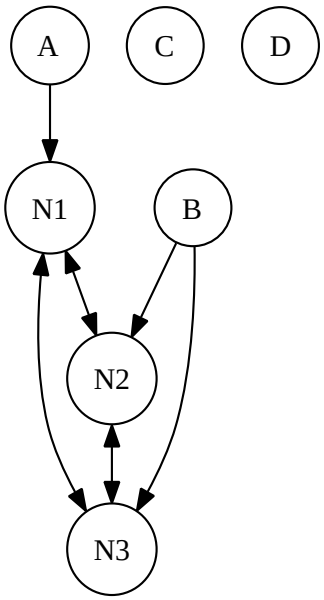


Figure 6.66: Network Learnt by the LUMIN Learner for Second Causal Loop Network with $NMI = 0.035, NCMI = 0.9$

headed arrows. The high value of NCMI suggests that the double headed arrows are not caused by finding weak indications of causal direction, but rather that there are strong indications that causality appears to flow both ways. We have to admit that initially this surprised us, but in retrospect this is true of causal loops, without explicit timing information it could, and probably should, appear that causality was bi-directional.

6.7.3 Third Causal Loop Test Network

Figure 6.67 shows the graph used for the third causal loop test network. As previously the

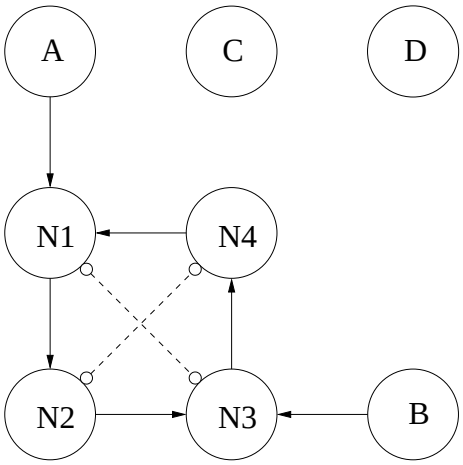


Figure 6.67: Third Causal Loop Test Network

dashed lines indicate likely relationships which will exist due to the deterministic nature of the

relationships in the graph. A 20000 record dataset was generated using a Gaussian distribution. The grow shrink learner produced the graph shown in figure 6.68. The learner reported an issue

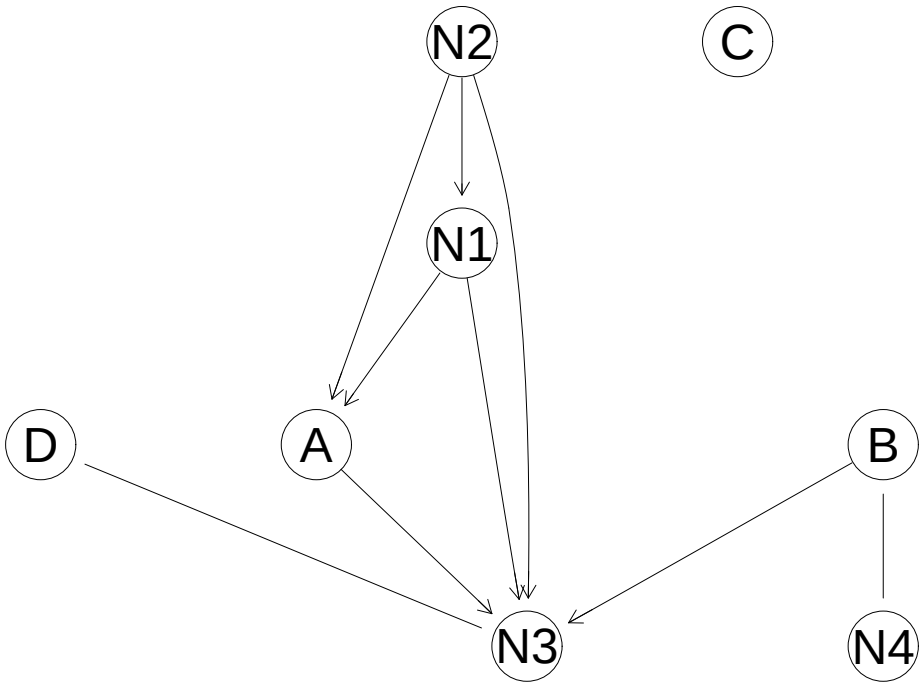


Figure 6.68: Network Learnt by Grow-Shrink Learner for Third Causal Loop Network

with the $N3 \rightarrow B \leftarrow N4$ triplet with incompatible arc directions. This is a little odd as that triplet does not form part of the loop and this may therefore be a problem unrelated to the loop itself. The hill climbing learner produced the graph shown in figure 6.69. The max-min hill climbing learner produced its best graph with the target nominal type 1 error rate of the conditional independence test at a value of 0.9. It produced the graph shown in figure 6.70. Lastly LUMIN produced the graph shown in figure 6.71 with an NMI value of 0.012 and an NCMI value of 0.75. The link between the variables A and $N2$ appears to be an artifact as although there was sufficient NMI between them to create a link, there was insufficient NCMI when examining the link to determine any likely direction of causal influence.

6.7.4 Causal Loop Results and Analysis

So far we have simply presented the graphs learnt by the various learners when dealing with causal loops without trying to provide a direct measure of comparison. It is difficult to determine a fair comparison for these learnt networks as one feature that should exist is a loop which the learners other than LUMIN are designed not to produce. Also as previously noted the direction of causality between the variables in the loop could be seen as going in either direction. So, in

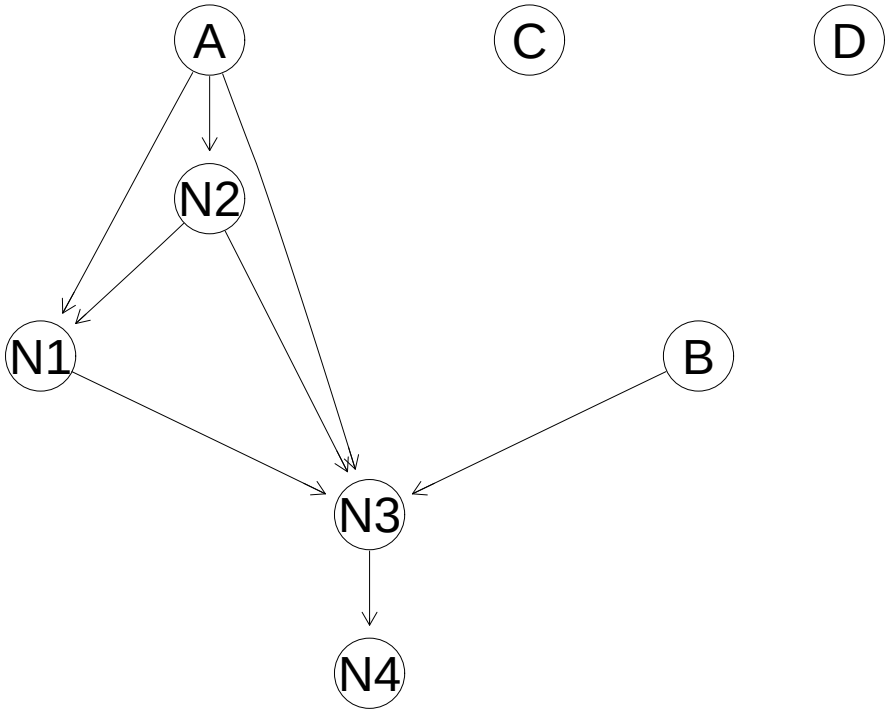


Figure 6.69: Network Learnt by Hill Climbing Learner for Third Causal Loop Network

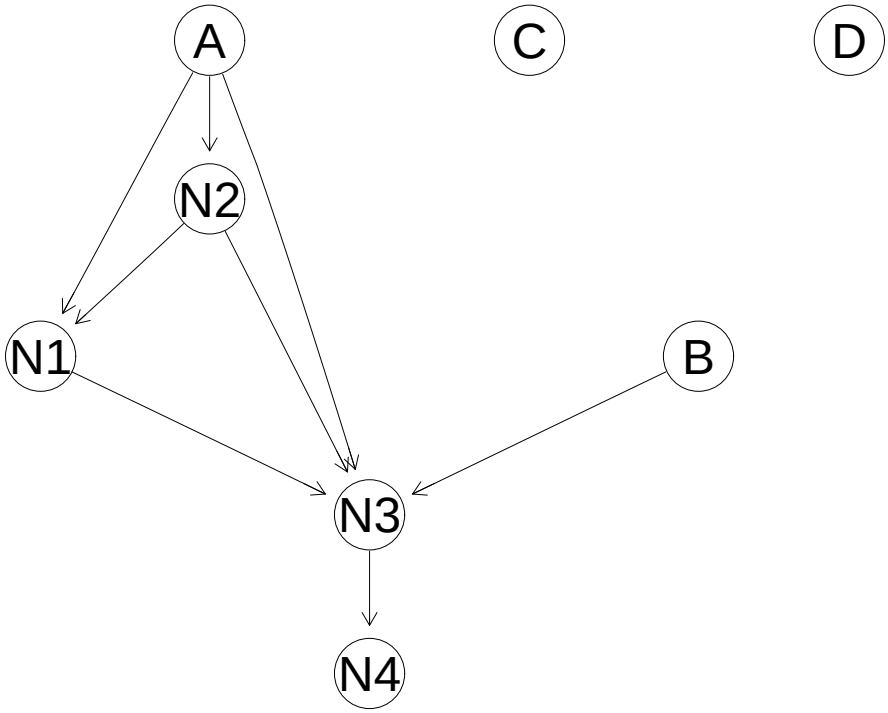


Figure 6.70: Network Learnt by Max-Min Hill Climbing Learner for Third Causal Loop Network with $\alpha = 0.9$

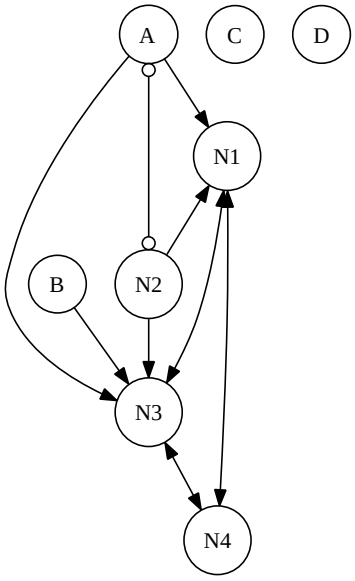


Figure 6.71: Network Learnt by the LUMIN Learner for Third Causal Loop Network with $NMI = 0.012, NCMI = 0.75$

order to produce a simple comparison we will use this variation on the CL metric with regards to the relationships between variables in the loop:

- Relationships between variables in the network indicated in the test network diagram with a dashed line will be ignored.
- Any of $\leftarrow, \rightarrow$ or $\longleftrightarrow$ will be considered to be correct, but if there is no actual loop the score will be penalised by 2. The very small penalty is so as not to bias the results too much in LUMIN's favour.

This will allow a direct comparison between learners where there is only a small penalty for failure to detect a loop as a large penalty would unfairly bias the score in favour of LUMIN. Using this modified version of the CL metric we can now score the various learnt networks. Table 6.16

	First	Second	Third
Grow-Shrink	8	18	22
Hill Climbing	12	22	15
Max-Min Hill Climbing	8	18	15
LUMIN	6	4	6

Table 6.16: Modified CL Metric Points Lost by Learners on Causal Loop Test Networks (small values are better)

gives the scores in terms of lost points for the learnt causal loop networks. The LUMIN learner performs well in this series of tests in general outperforming the other learners by more than the

2 points allowed for the discovery of the loop. This may be because not allowing for a loop has consequences for other decisions the learners make about the overall network structure. This idea may be borne out by the result of the third test network, for this network LUMIN like the other learners failed to find the required $N1 \rightarrow N2 \rightarrow N3 \rightarrow N4 \rightarrow N1$ loop, but still managed a significantly better score than the other learners. Perhaps this was because it accepted a different loop $N1 \rightarrow N3 \rightarrow N4 \rightarrow N1$ as part of the solution. The results are shown in graphical from in figure 6.72.

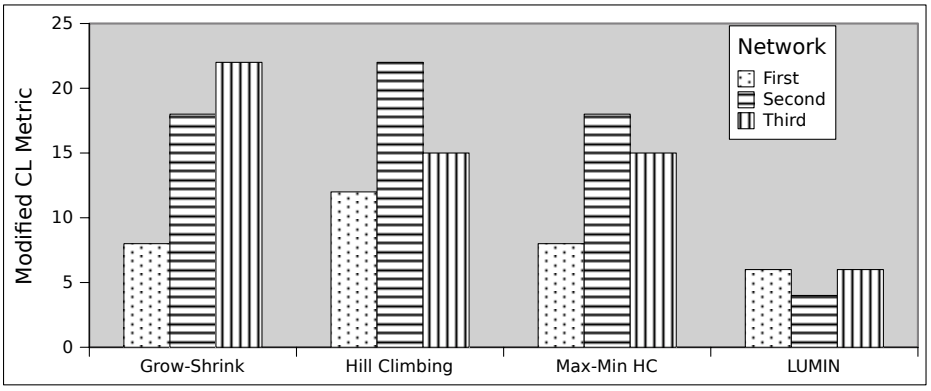


Figure 6.72: Graph of the Performance of all Learners on the Causal Loop Test Networks (smaller CL values are better)

6.8 Summary

The six standard datasets analysed show LUMIN to be competitive with the other learners. Given reasonable values for its NMI and NCMI thresholds LUMIN produces a good approximation of the original network. Under the test conditions LUMIN produced the most accurate ALARM network for both datasets. The Hailfinder network is much more of a challenge for constraint-based learners in general and LUMIN in particular, as it has many single parent nodes and many of those share the same parent. Even so while the Hill Climbing learner did best, LUMIN scored better than both Grow-Shrink and MMHC learners on both datasets. With the Insurance network we added the use of some domain information to test this feature of the LUMIN learner. Using limited domain information LUMIN outperformed the other learners. We believe these tests show that LUMIN has succeeded in performing its basic task of recovering causal structure from data, and does so in a manner that is competitive with other current causal learners.

The networks constructed to test LUMIN with non-linear relationships, demonstrate that it is at least comparable to the other learners, and in the case of trigonometric relationships superior

to most of them. While the other learners are able to cope with some non-linear relationships, often their underlying assumptions do not include them. The fairly simple test cases suggest that LUMIN would be a good choice for analysis in cases where non-linear relationships are expected.

The causal loop networks test an area where LUMIN is different by design to most other causal learners, that is, LUMIN was designed to allow for causal loops. The testing revealed that when a loop exists, unless there is specific data that allows the determination of the direction of the loop, we would expect the links in the loop to appear as bi-directional. This is something which is clear in hindsight, but we did not consider this before working on examples of causal loops. By implication to discover loops it is important to allow some form of undirected or bi-directed link, or alternatively to specify timing information, which is not something all causal learners allow. The causal loops example also suggest that the inability to deal with causal loops can detrimentally effect the learners ability to correctly learn the other non-loop parts of the network which interact with the loop.

Chapter 7

Conclusions

This chapter looks at what has been achieved, whether we met our goals and what can be improved on in future. We start by looking at LUMIN's ability to discover causal relationships. In section 7.2 we look at how LUMIN can use domain information. Section 7.3 examines ways in which both the current implementation and the underlying theory can be developed. Lastly we summarise the conclusions about causal discovery and the LUMIN learner.

7.1 Relationship Discovery

The basic goal of the LUMIN program was to be able to discover the existence and direction of causal influence in causal relationships where the only data was an observational dataset.

7.1.1 Basic Causal Discovery

The results with the ALARM, Hailfinder and Insurance networks in sections 6.3, 6.4 and 6.5 show that, at least on these standard datasets, LUMIN is competitive with a number of other causal learners in this basic aspect of causal discovery[1]. These datasets show some variation in the abilities of the different learners to deal with differing situations. We saw that the Hailfinder dataset posed a number of problems particularly with the number of single parent children. However, while the performance of the hill climbing learner was significantly better than the LUMIN learner on the Hailfinder datasets, it is still true that LUMIN was competitive with the other learners on these datasets.

[1] We have also had success with a number of other network not included in this thesis.

7.1.2 Non-Linear Relationships

One of our goals was to allow for the discovery of the widest possible range of relationship types. Many current causal learners make assumptions about the form of the relationships, linear, Gaussian, acyclic etc. LUMIN makes only the assumption that causally related variables will share mutual information and that conditioning on a third variable can be used to help determine the direction of causal relationships. Since MI is largely insensitive to the form of the underlying relationship LUMIN should be able to discover many different types of causal relationship. We constructed three simple networks, and a fourth that was just the combination of the three, using non-linear relationships, and tested the learners with datasets from these networks, see section 6.6. The results for these simple networks show that LUMIN is competitive with the other learners on these relationships. With the trigonometric relationships LUMIN was significantly better at determining the causal relationships than the other learners.

7.1.3 Causal Loops

Many causal learners assume the causal Markov condition, see definition 20, section 2.9.3. While this can be useful in constructing a network it prevents the learning of causal loops. One of our goals for LUMIN was to be able to learn causal loops. We constructed three simple networks each of which contained a loop. Section 6.7 details the networks and the results of the various learners when working with datasets produced by them. The three learners we have been using for comparison, grow-shrink, hill climbing and max-min hill climbing all assume the causal Markov condition. So, LUMIN was the only learner which might possibly learn the loops in the networks. We modified the CL metric slightly to allow for the uncertainty in the direction of loops, we don't actually supply enough information for determination of loop direction, and add a small penalty for failure to detect a loop. The results show that LUMIN can determine the presence of a causal loop, although it does not always do so. Further it appears, both from the simple test networks and more complex ones not included in the thesis, that inability to learn a loop can reduce the accuracy with which the other parts of the network, that interact with the loop, are learnt.

7.2 Using Domain Information

One of our aims was to allow the learning system to make use of any available domain information relating to causal influences between the variables being considered. Many learners can use prior information to help guide their operation, but often the form of information required, such as prior probabilities, or an initial network, is not easy for a human expert to construct. We wanted the LUMIN program to allow a simple presentation of domain information. The LUMIN system is very flexible in this respect as it allows:

- Predetermining the direction of causality between two variables if they appear to be causally related
- Specifying that two variables are causally related and allowing the normal mechanism to determine the direction of causality
- Specifying that two variables are causally related and the direction of their causality
- Forbidding a causal relationship between two variables.

Thus with regard to pairs of variables any prior domain knowledge can be incorporated in the learning process. Domain information is used to alter the initial network LUMIN constructs before it attempts direction determination and link removal and so acts as an input to the learning process rather than simply altering the final result. The method for specifying the likely direction of causal influence is very flexible allowing each variable to belong to a number of different domain groups and its direction of causal influence with another variable to be determined by their common domain(s). Inconsistent domains are not allowed, that is if two variables share two or more domains which define opposite directions of causal influence then the program will terminate with an error. In addition to determining the direction of causal influence if a relationship is found to exist, it is possible to specify the existence of a relationship and, if required, its direction. As before this is done prior to the normal link direction analysis and will therefore influence it. Lastly it is possible to forbid certain relationships, thus even if the initial NMI test suggested the presence of a link, the link would be removed before the analysis commenced. This level of flexibility allows for testing of 'what if' scenarios, something that is not possible with all learners.

The tests using datasets generated by the Insurance network, see section 6.5, show how domain information can improve the performance of the LUMIN learner. In this instance the only

domain information provided was to help determine link direction between some variables if a link was found, no links were specified and none were forbidden. In otherwise directly comparable examples supplying some simply reasoned domain information improved the performance of the learner, in terms of the loss of points judged by the Causal Likeness metric, from 164 to 132 and from 155 to 133 points lost respectively.

7.3 Future Research

While we have shown that the current implementation of the LUMIN learner works reasonably well and is competitive with other similar learners there are many areas in which it could potentially be improved.

7.3.1 Link Removal

In its basic form LUMIN produces graphs with an excessive number of links between the variables, this is to be expected as it follows logically from the basic assumptions behind the learner. If a variable A is a cause of a variable B which is in turn the cause of a variable C, then we would expect A to share mutual information with both B and C. The current heuristic link removal algorithms do a reasonable job of removing unwanted links, and are themselves not to computationally onerous. The heuristic link removal algorithms are based on what seemed to be a reasonable interpretation of what we would expect to see in various circumstances, and appear to work reasonably well. However, it is known that they are not perfect and a better understanding of the basic science involved may allow for better performance in the removal of extraneous links.

Sensitivity analysis might help determine links which appear to have little value, that is which links can be removed without any knock on effect in the remainder of the network. At present the network is otherwise unchanged after unwanted links have been removed. However, since the removed links could have been involved in the determination of direction for other links it might be worthwhile investigating redetermining the direction for some or all of the remaining links. Currently all links are initially considered equal when it comes to considering which links to remove. It would be possible to use the strength of a link, that is the amount of mutual information it represents, as a guide and to look to removing weaker links first.

7.3.2 Threshold Values and Tests

Using the learner at present requires choosing two threshold values one for normalised mutual information to determine link inclusion, and another for normalised conditional mutual information to determine the direction of causal influence. It may be possible to automate selection of *optimal* values for these thresholds for any given database.

The current thresholds are one-tailed tests, that is, the result is either below or above some defined value. Moving to two tailed tests, by simply having lower and upper thresholds, perhaps using sensitivity tests to determine inclusion of those items that fall between the two thresholds, might improve the learner's performance. The amount of mutual information that indicates a significant relationship might be expected to vary between variables. If a variable has a single cause, then potentially it would share a great deal of mutual information with it. However, suppose a variable has fifty other variables which all influence it directly. It is likely that, in any given dataset, it shares little mutual information with many of its causes. It may be possible to use sensitivity analysis to help determine on a per variable pair basis suitable thresholds for significant mutual information.

7.3.3 Leveraging Link Strength

Currently link direction determination uses all the available information, that is, every triplet of which a link is a member is used to help determine its direction. If different directions are determined then the link is made bi-directional. It may be advantageous to use both the strength of the links in a triplet and the value of the normalised conditional mutual information test when determining the probable link direction. This may allow retaining a single direction even when both directions are indicated if the strength of support for one direction is significantly greater than that for the other.

7.3.4 Hidden Variables

The current implementation of LUMIN makes no specific allowance for hidden variables. While, in general, hidden variables should not lead to an incorrect graph[2] it may be advantageous to attempt to identify where hidden variables may interact with those which are observed. Kwoh and Gillies, [Kwoh & Gillies 1996], indicated that the performance of a BN could be improved by adding hidden variables in cases where a conditional independence entailed in a network was

[2] Albeit with some variables missing.

not observed in the data. This may be a useful area of future research particularly as the NCMI test used by LUMIN could be substituted for conditional independence.

7.4 Summary

LUMIN has demonstrated basic causal discovery competitive with other causal learners, an ability to deal with non-linear causal relationships and the ability to discover and represent causal loops. The basic LUMIN algorithm has a computational complexity that is polynomial in both the number of variables and the number of records. LUMIN can make use of a wide variety of domain information and does so in a manner that works with and enhances its normal learning abilities. LUMIN uses all the data it has available, when it is given incomplete records it uses the data it has and only skips incomplete records when working with variables for which that record has no value. The input data format is a variation of that used by the MLC++ learning suite and the output format is that of the Dot language from the Graphviz suite. So, both input and output formats are simple text based and, at least in their basic form, well documented. Lastly while the current implementation of LUMIN appears to be competitive with other causal learners, there are a number of ways we believe its performance can be improved.

We believe the LUMIN learner has fulfilled the required capabilities specified in our original design, see section 5.2. There are a number of areas in which we believe the LUMIN learner can be improved as outlined in section 7.3, but we hope LUMIN will be a useful addition to the available causal learners, particularly when working with observational datasets.

Appendix A

Publication

During the course of the thesis the following paper was written which examines the behaviour of a number of learners when used to predict the outcome of football matches. An expert provides domain information which is used to construct a BN. This is then compared with a number of other learners, including a learnt BN, and the accuracy of their predictions is tested.

ARTICLE IN PRESS

Available online at www.sciencedirect.com

SCIENCE DIRECT®

Knowledge-Based Systems xxx (2006) xxx–xxx

Knowledge-Based SYSTEMS

www.elsevier.com/locate/knosys

Predicting football results using Bayesian nets and other machine learning techniques

A. Joseph *, N.E. Fenton, M. Neil

Computer Science Department, Queen Mary, University of London, UK

Received 21 April 2005; accepted 6 April 2006

Abstract

Bayesian networks (BNs) provide a means for representing, displaying, and making available in a usable form the knowledge of experts in a given field. In this paper, we look at the performance of an expert constructed BN compared with other machine learning (ML) techniques for predicting the outcome (win, lose, or draw) of matches played by Tottenham Hotspur Football Club. The period under study was 1995–1997 – the expert BN was constructed at the start of that period, based almost exclusively on subjective judgement. Our objective was to determine retrospectively the comparative accuracy of the expert BN compared to some alternative ML models that were built using data from the two-year period. The additional ML techniques considered were: MC4, a decision tree learner; Naive Bayesian learner; Data Driven Bayesian (a BN whose structure and node probability tables are learnt entirely from data); and a *K*-nearest neighbour learner. The results show that the expert BN is generally superior to the other techniques for this domain in predictive accuracy. The results are even more impressive for BNs given that, in a number of key respects, the study assumptions place them at a disadvantage. For example, we have assumed that the BN prediction is 'incorrect' if a BN predicts more than one outcome as equally most likely (whereas, in fact, such a prediction would prove valuable to somebody who could place an 'each way' bet on the outcome). Although the expert BN has now long been irrelevant (since it contains variables relating to key players who have retired or left the club) the results here tend to confirm the excellent potential of BNs when they are built by a reliable domain expert. The ability to provide accurate predictions without requiring much learning data are an obvious bonus in any domain where data are scarce. Moreover, the BN was relatively simple for the expert to build and its structure could be used again in this and similar types of problems.

Keywords: Bayesian nets; Machine learning; Football

1. Introduction

Bayesian networks [1], BNs, provide a means for capturing, displaying, and making available in a usable form knowledge, often obtained from experts in a given field. This knowledge is often obtained from experts and can be based on subjective judgements as well as (or even instead of) data. Predicting the outcome of a football match is an ideal application (although it is far removed from other applications we have been involved with such as [2,3,5]). It is in just this type of problem, with many complex interacting factors, that BNs excel. It is possible for a domain expert, in collaboration with a BN expert, to construct a network detailing the important relationships between the factors involved, and the node probability tables, (NPTs). In this paper, we look at the performance of an expert constructed BN in predicting the outcome (win (2), lose (0), or draw (1)) of matches played by Tottenham Hotspur ('Spurs'). The BN was originally developed at the start of the 1995–96 season. Since, it involves specific players, the model was only relevant for two seasons (after which some of the key players were no longer at the club). Hence, the study is restricted to all league matches played by Spurs during the two consecutive seasons 1995/1996

* Corresponding author. Tel.: +44 020 8597 9578; fax: +44 020 8980 6533.

E-mail addresses: adrianj@dcs.qmul.ac.uk (A. Joseph), norman@dcs.qmul.ac.uk (N.E. Fenton), martin@dcs.qmul.ac.uk (M. Neil).

URL: http://www.dcs.qmul.ac.uk/researchgp/radar/ (A. Joseph).

0950-7051/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.knosys.2006.04.011

and 1996/1997. So why, almost 10 years after the expert BN was developed, have we returned to this particular problem? It is because we had a unique opportunity for a direct comparison between the expert BN and a range of alternative ML models. Such studies are relatively rare and the results and lessons learnt should be of interest to researchers outside of this particular domain (even those readers who have no interest in Spurs or football in general). The performance of the expert BN model is compared with four alternative machine learning (ML) models:

- A naive BN.
- A BN learnt from statistical relationships in the data [6].
- A *K*-nearest neighbour implementation [7].
- A decision tree [8].

The aim was to see how the expert constructed BN compares in terms of both predictive accuracy and explanatory clarity for the factors effecting the result of the matches under investigation.

Section 2 discusses the issues of model setup and how we selected the football match data to learn from. Section 3 is a brief explanation of the learning techniques used and our approach to the analysis. Section 4 provides the results of the learners for each of the data sets used, while Section 5 provides a summary of the predictive accuracy. Section 6 summarises our conclusions and looks at some possible directions of future work.

2. Selecting relevant information

There are a large number of factors which could effect the outcome of a football match from the perspective of one of the teams involved. One of the difficulties in any investigation of the relationships involved in a given effect is that to a large extent the assumption of a particular model determines the attributes to study and predetermines the possible relationships that can be found. So, the act of choosing which model and attributes to study sets a boundary on what can be discovered.

2.1. Constructing an initial model

When approaching a new problem there are two techniques which are commonly used. The first assumes we have some idea how the situation under investigation works, construct a model, and using this model select the attributes believed to contribute to the effect under investigation. An example of this approach to this type of problem is given in [9]. The second approach assumes little knowledge of the underlying mechanisms involved so we look at all the *probably* relevant attributes and try to determine those which have the most significant effect. This is still in effect the construction of an a priori model, but only a very informal one. In this paper, we take the second approach.

2.2. The expert model

The expert BN (see Fig. 1) uses only a few features:

- The presence or absence of three players, Sherringham, Anderton, and Armstrong. So in each match each of these values was true or false.
- The playing position of Wilson represented by him playing in midfield or not.
- The quality of the opposing team. This particular variable was measured on a simple 3-point scale (high, medium, and low). Although based on expert judgement, it matches closely with the teams' final league positions ('top 6', 'middle 8', or 'bottom 6') and so would appear to be an accurate reflection of their average performance.
- Venue (whether the game is played at Spurs' home ground or away).

The BN shows how the expert constructed the relationships between the chosen factors and the outcome of the game. In addition to the result node (win, lose, or draw) the BN includes three other nodes to simplify the structure:

- **Attack** which represents the quality of the Spurs attacking force (low, medium, and high).
- **Spurs_quality** the overall quality of the Spurs team (low, medium, and high).
- **Performance** how well the team will perform given their own quality and that of the opposition (low, medium, and high).

2.3. The general model and its known weaknesses

We allowed the machine learners to use both the same and an alternate set of features compared to the expert BN. Specifically, the initial set of factors were the basic factors in the expert model, plus all the other registered Spurs' players (as playing or not playing) rather than just the four 'special' players in the expert BN minus the playing position of Wilson. The particular values for Opposition quality in each game were the same as those used by the expert BN.

During a game players can be injured, substituted, be sent off, or have their playing positions changed. The solution chosen to deal with these issues was to use the information about only those players who started the game. Similarly Wilson's playing position could change during the course of the match, only his initial playing position was considered.

In general terms this problem is not particularly easy from a machine learning perspective. There is not much data to go on. We have the results of two seasons' games, a total of 76 matches and for the general model a total of 30 attributes, (28 players, venue, and opponent quality). There were changes to the Spurs' squad during this period. The simple convention of a player either playing or not was chosen to avoid having missing data entries with regards to

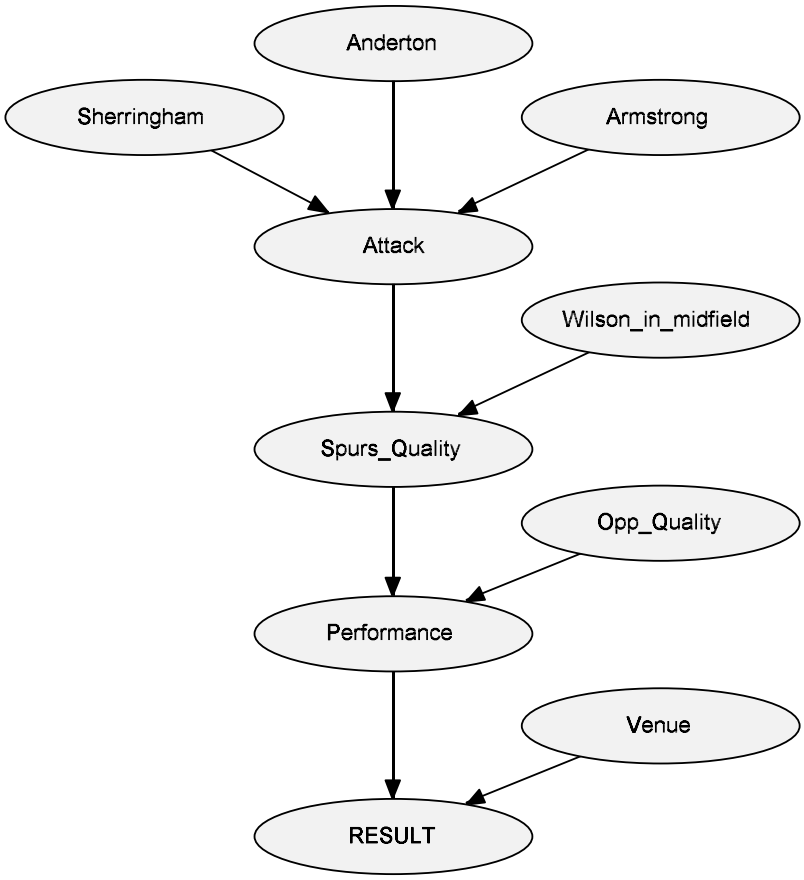


Fig. 1. Expert constructed BN for Tottenham Hotspur's performance.

squad changes. There are, of course, other external factors which effect the outcome of a game. So, even in the best case we expect to have noise in the data. Since players, except Wilson, are only considered from the point of view of playing or not playing, the effect of any player who was always present will be ignored. This is because the learners can only compare the difference in the outcome of matches with a player present or absent.

It is also worth noting that all the models (including the expert BN) are inherently asymmetric. Whereas for Spurs we consider the particular players involved in any given match to be significant, for all their opponents we only have a general rating for their overall quality.

3. Machine learning techniques and our analysis assumptions

There are a large number of ML techniques each with different strengths and weaknesses. Choosing which is the most appropriate technique often requires an understanding of both the problem domain and the different learning methods. A good introduction to many machine learning techniques can be found in [10]. The machine learners used in this analysis were:

MC4 Decision trees. Decision trees provide a visual representation of relationships which appear to effect the situation under investigation. Pruning is generally used to reduce the size of the tree. The confidence method of pruning was used.

Naive Bayesian learner. The Naive Bayesian learner makes the simplifying assumption that all the attributes are independent.

Data Driven Bayesian learner. The complex Bayesian learner as implemented by Hugin attempts to learn the structure of the network by looking at the correlation between the attributes. Once the structure has been determined data can then be used to determine the node probability tables. The strength of a correlation required to trigger the joining of two nodes can be adjusted.

Expert constructed Bayesian network. When expert knowledge of a given domain is to be represented as a BN the usual process is for the domain expert(s) and BN expert(s) to jointly construct the BN. If sufficient data are available then the NPTs can be directly learnt and then adjusted if required. However, when there is insufficient data to learn the NPTs these must also be obtained from the expert(s).

***K*-nearest neighbour.** *K*-nearest neighbour learners use a *likeness* approach to prediction. That is, they look at the instances most like the test case and usually have some voting method by which the prediction is

Table 1
Comparison of learner accuracy with expert model data

Train period–Test period	Number of correct predictions by learner					
	Most common	MC4	Naive BN	Hugin BN	Expert BN	KNN
95/96–95/96 season	16 (42.11%)	28 (73.68%)	26 (68.42%)	21 (55.26%)	20 (52.63%)	37 (97.37%)
96/97–96/97 season	18 (47.37%)	30 (78.95%)	31 (81.58%)	26 (68.42%)	25 (65.79%)	37 (97.37%)
Average for full seasons	17 (44.74%)	29 (76.32%)	28.5 (75.00%)	23.5 (61.84%)	22.5 (59.21%)	37 (97.37%)
Period 1–period 234 95/96	12 (42.86%)	8 (28.57%)	9 (32.14%)	8 (28.57%)	14 (50.00%)	12 (42.86%)
Period 12–period 34 95/96	7 (38.89%)	6 (33.33%)	6 (33.33%)	3 (16.67%)	10 (55.56%)	7 (38.89%)
Period 123–period 4 95/96	2 (25.00%)	2 (25.00%)	2 (25.00%)	2 (25.00%)	3 (37.50%)	2 (25.00%)
Sum for 1995/1996 periods	21 (38.89%)	16 (29.63%)	17 (31.48%)	13 (24.07%)	27 (50.00%)	21 (38.89%)
Period 1–period 234 96/97	11.5 (41.07%)	10 (35.71%)	13 (46.43%)	11 (39.29%)	19 (67.86%)	11 (39.29%)
Period 12–period 34 96/97	7.5 (41.67%)	7 (38.89%)	10 (55.56%)	3 (16.67%)	10 (55.56%)	5 (27.78%)
Period 123–period 4 96/97	5 (62.50%)	2 (25.00%)	5 (62.50%)	2 (25.00%)	3 (37.50%)	1 (12.50%)
Sum for 96/97 periods	24 (44.44%)	19 (35.19%)	28 (51.85%)	16 (29.63%)	32 (59.26%)	17 (31.48%)
Period 23 95/96–period 4/1 95/97	6 (33.33%)	4 (22.22%)	6 (33.33%)	Unavailable	9 (50.00%)	7 (38.89%)
Period 234 95/96–period 1 96/97	4 (40.00%)	2 (20.00%)	4 (40.00%)	3 (30.00%)	6 (60.00%)	3 (30.00%)
Period 34 95/96–period 12 96/97	8 (40.00%)	6 (30.00%)	8 (40.00%)	11 (55.00%)	15 (75.00%)	7 (35.00%)
Period 4 95/96–period 123 96/97	6 (20.00%)	8 (26.67%)	6 (20.00%)	10 (33.33%)	22 (73.33%)	8 (26.67%)
Period 4/1 95/97–period 23 96/7	6.67 (33.33%)	7 (35.00%)	8 (40.00%)	7 (35.00%)	16 (80.00%)	7 (35.00%)
Season 95/96–season 96/97	13 (34.21%)	8 (21.05%)	13 (34.21%)	20 (52.63%)	25 (65.79%)	15 (39.47%)
Sum for cross season periods	43.67 (32.11%)	35 (25.74%)	45 (33.09%)	51 (43.22%)	93 (68.38%)	47 (34.56%)
Overall average percentage	40.05%	41.72%	47.86%	39.69%	59.21%	50.58%
Overall disjoint training/data	38.48%	30.19%	38.81%	32.31%	59.21%	34.98%

chosen. The usual measure of *likeness* is Euclidean distance as plotted on an *n*-dimensional graph where each dimension is one of the supplied attributes.

All the learners used were part of the MLC++ [11] package[1] apart from the complex Bayesian learner which was part of the Hugin tool[2], the Hugin tool was also used to *run* the expert constructed BN.

The different models do not all provide the same sort of prediction. The MC4 and KNN learners usually give a prediction in the form of an unqualified value from the possible range of values. BNs do not make predictions in the same format as the MC4 or KNN learners. Rather than supply a simple answer they supply a probability for each of the possible outcomes. This allows for a greater sensitivity of prediction; the BN not only makes a prediction, but is also able to provide some idea of confidence in the prediction. To make a direct comparison with the learners we had to interpret the BN prediction as a definite result (win, lose, or draw). Our approach was to choose the result with the highest predicted probability, irrespective of how close two or more results might be. In cases where two or more of the outcomes of the BN were equally likely we deemed that the prediction was incorrect (even if the actual result was one of the two most likely). This approach clearly treats BNs harshly in the analysis. In reality, a prediction involving equal (or nearly equal) probabilities would be useful. For example, if we were betting on the outcome of a game, and the BN predicted Win 45% Draw 45% Loss 10% then this would indicate a likely win for an each way bet. However, such an analysis of the potential value of a shared highest probability prediction is beyond the scope of this paper.

We divided the match data into disjoint subsets so that some could be used for training and separate data used to check the accuracy of the learners. The data for each season was divided up into three groups of ten matches and one group of eight matches, organised chronologically. We maintain the ordering of games and always organise the training so that the training data set are chronologically immediately before the test data set. For comparison we also used each complete season's data for training and test set for the learners. This again prejudices the results against the expert BN because this will tend to overestimate the accuracy of all the other learners. The machine learners were tested with both our general model data and with the data used by the expert BN. Using the two data sets allows for a direct comparison with the same, expert chosen, data set and a more general comparison with a data set a non expert might choose. The results for both the general data and the expert chosen data, shown in Tables 1 and 2, are similar. Where changes in classification error are mentioned they are relative to the error obtained by choosing the most common result from the training data.

4. Results analysis

In this section, we compare the accuracy of the different models' predictions (for some general information on making comparisons between learners see [12]). We also

[1] Version 2.01 of the MLC++ libraries was used, modified to run under the GNU/Linux operating system. All the MLC++ learners were used with their default settings except where noted otherwise.

[2] Version 6.1 of this tool was used for this paper.

Table 2
Comparison of learner accuracy with expert model data

Train period–Test period	Number of correct predictions by learner					
	Most common	MC4	Naive BN	Hugin BN	Expert BN	KNN
95/96–95/96 season	16 (42.11%)	25 (65.79%)	22 (57.89%)	23 (60.53%)	20 (52.63%)	27 (71.05%)
96/97–96/97 season	18 (47.37%)	26 (68.42%)	25 (65.79%)	26 (68.42%)	25 (65.79%)	32 (84.21%)
Average for full seasons	17 (44.74%)	25.5 (67.11%)	23.5 (61.83%)	24.5 (64.47%)	22.5 (59.21%)	29.5 (77.63%)
Period 1–period 234 95/96	12 (42.86%)	8 (28.57%)	7 (25.00%)	8 (28.57%)	14 (50.00%)	9 (32.14%)
Period 12–period 34 95/96	7 (38.89%)	5 (27.78%)	9 (50.00%)	0 (0.00%)	10 (55.56%)	8 (44.44%)
Period 123–period 4 95/96	2 (25.00%)	4 (50.00%)	3 (37.50%)	2 (25.00%)	3 (37.50%)	4 (50.00%)
Sum for 1995/1996 periods	21 (38.89%)	17 (31.48%)	19 (35.19%)	10 (18.52%)	27 (50.00%)	21 (38.89%)
Period 1–period 234 96/97	11.5 (41.07%)	11 (39.26%)	12 (42.86%)	13 (46.43%)	19 (67.86%)	7 (25.00%)
Period 12–period 34 96/97	7.5 (41.67%)	6 (33.33%)	8 (44.44%)	6 (33.33%)	10 (55.56%)	8 (44.44%)
Period 123–period 4 96/97	5 (62.50%)	4 (50.00%)	2 (25.00%)	2 (25.00%)	3 (37.50%)	3 (37.50%)
Sum for 1996/1997 periods	24 (44.44%)	21 (38.89%)	22 (40.74%)	21 (38.89%)	32 (59.26%)	18 (33.33%)
Period 23 95/96–period 4/1 95/97	6 (33.33%)	7 (38.89%)	7 (30.89%)	7 (30.89%)	9 (50.00%)	8 (44.44%)
Period 234 95/96–period 1 96/97	4 (40.00%)	7 (70.00%)	3 (30.00%)	6 (60.00%)	6 (60.00%)	5 (50.00%)
Period 34 95/96–period 12 96/97	8 (40.00%)	14 (70.00%)	9 (45.00%)	11 (55.00%)	15 (75.00%)	11 (55.00%)
Period 4 95/96–period 123 96/97	6 (20.00%)	6 (20.00%)	8 (26.67%)	4 (13.33%)	22 (73.33%)	7 (23.33%)
Period 4/1 95/97–period 23 96/97	6.67 (33.33%)	6 (30.00%)	8 (40.00%)	6 (30.00%)	16 (80.00%)	8 (40.00%)
Season 95/96–season 96/97	13 (34.21%)	22 (57.89%)	13 (34.21%)	21 (55.26%)	25 (65.79%)	14 (36.84%)
Sum for cross season periods	43.67 (32.11%)	62 (45.59%)	48 (35.29%)	55 (40.44%)	93 (68.38%)	53 (38.97%)
Overall average percentage	40.05%	45.77%	42.26%	40.58%	59.21%	47.21%
Overall disjoint training/data sets	38.48%	38.65%	35.74%	32.62%	59.21%	37.06%

look at any information provided by each model about the factors effecting the outcome of the games. Note that, because of space limitations, we do not include the full set of data and models. This is, however, all available online here [4].

4.1. The MC4 Learner

Decision tree learners like MC4 are good at dealing with relatively static situations, that is, situations in which the relationships between the various attributes are fixed. We were not sure how true this was of the Spurs team, and its performances, over the period being examined. The overall classification error of the MC4 learner for disjoint training and test data sets in the general model was 69.81% and 61.35% for the expert chosen data.

4.1.1. Complete seasons

The basic tree produced by MC4 when looking at the general model data for the 1995/1996 season is a fairly simple tree using only 6 of the available 30 attributes, the players Dozzell, Campbell, and Nethercott, the venue and the opposing team ranking. The tree, Fig. 2, shows Dozzell as a key player[3]. For the 1995/1996 season the MC4 analysis gives a reduction in the classification error of 34.57% and 23.68% for the general and expert models, respectively.

[3] It is interesting to note that after seeing this analysis the expert stated that while he suspected Dozzell was a key player this was not the general opinion at that time and he thus left Dozzell out of the expert BN.

An analysis of the 1996/1997 seasons matches produced a slightly more complex tree (which can be seen in [4]), using 8 rather than 6 attributes. MC4 analysis gives a reduction in the classification error of 31.58% using the general model and a reduction of 21.05% using the expert chosen data.

4.1.2. Separate training and test data – single season

The performance of the MC4 learner was, as expected, less impressive when it was only given part of a season's data and used to predict the remainder. The classification error for the tests using general model data from 1995/1996 season increased by 9.26%, and the same tests for the 1996/1997 season showed an increase in the error of 9.25%. The learner faired slightly better with the expert chosen data giving an increase in error of 7.41% and 5.55% for the 1995/1996 and 1996/1997 seasons, respectively. The performance of the learner did not seem to improve with increasing amounts of training data. The trees built by MC4 with increasing data develop towards that built with the full season's data.

The performance of the learner over all five cross season periods, for the general model, was quite poor. The classification error for the general model averaged over all the cross season tests increased by 6.37%. The learnt tree for the end of the 1995/1996 season, period 4, and the beginning of the 1996/1997 season, period 1, is the largest of the trees for any two period group. This may indicate that significant changes take place between seasons, which would not be contradicted by the slight drop in performance of cross season tests compared to similar intra-season tests. There is also a drop in the predictive

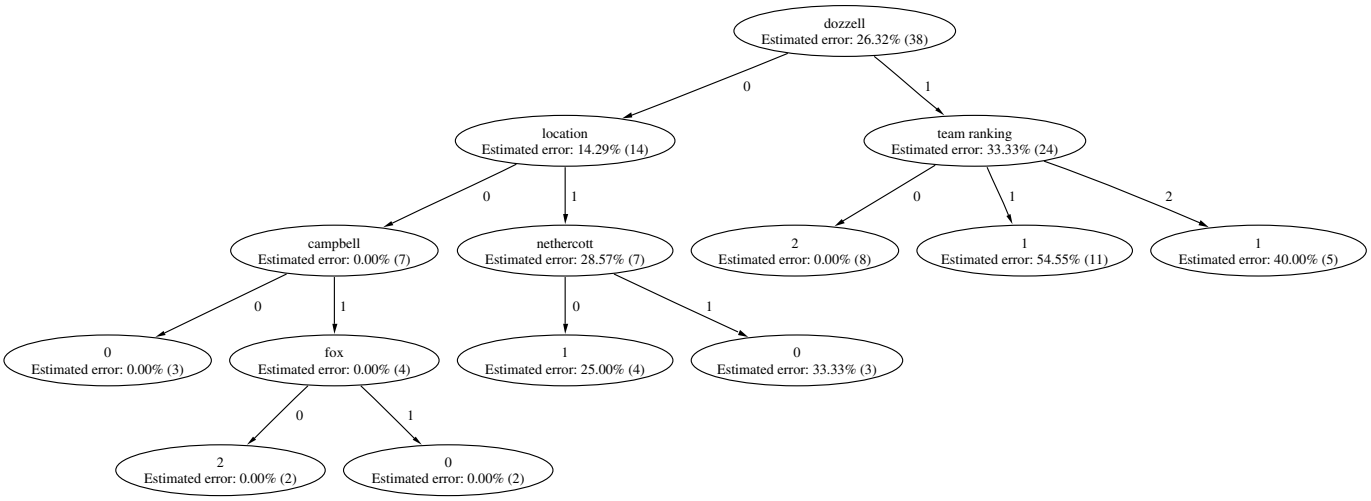


Fig. 2. Decision Tree for the general model 95/96 season with error estimates.

ability of the most common test result which means that overall for the cross seasons tests the classification error from the MC4 learner was 6.37% worse than that from choosing the most common test result. Over the same period the expert chosen data gave a better result with an average reduction in the error of 13.48%.

4.2. Naive Bayesian learner

While the attributes of the problem do not adhere to the strict independence assumption of the naive Bayesian learner we would expect there to be a reasonable match and thus for this learner to perform relatively well. This is reflected in that for non-overlapping training and test data sets on the general model this learner came second overall with a classification error of 61.19%. Interestingly on the expert chosen data the naive Bayesian learner only came in fifth best with a classification error of 64.26%.

4.2.1. Complete seasons

For the 1995/1996 season the Naive Bayesian learner correctly predicted the result of 26 and 22 of the 38 games in the general and expert models respectively. This is a reduction in the classification error of about 26.31% and 15.78%. The naive Bayesian classifier gives no direct indication of the importance of any given attribute. However, looking at the NPT for the classifier in the general model we can see that the six most significant attributes in descending order are: Team Ranking, Dozzell, Edinburgh, Anderton, Dumitrescu, and Calderwood. There is some, limited, agreement between MC4 and the naive Bayesian learner on the significant attributes, they agree on the two most important of the thirty attributes for the 1995/1996 season. For the 1996/1997 season the Naive Bayesian learner correctly predicted the result of 31 and 25 of the 38 games for the general and expert models, respectively. This is a reduction in the classification error of about 34.21% and 18.42%.

4.2.2. Separate training and test data – single season

The results for the 1995/1996 season showed the average classification error to be 7.41% and 3.70% higher for the general and expert data sets, respectively. However, for the 1996/1997 season the general model classification error was 7.41% lower while that for expert data set model increased by 3.70%. Most classifiers achieved better results for the 1996/1997 season than the 1995/1996 season which may indicate greater stability in the team in the later season.

4.2.3. Separate training and test data – cross seasons

The cross season results for the naive Bayesian learner were roughly comparable to its in-season results. Overall it achieved a classification accuracy of 33.09% and 35.29% for the general and expert models which only bettered the most common classifier by 0.98% and 3.18%, respectively. Ignoring the case using the same training and test data for the complete seasons, the naive Bayesian learner came out second best overall on the general model and fifth overall on the expert model.

4.3. Data driven Bayesian learner

The BNs for the data driven Bayesian learner were generated using the structural learning wizard from the Hugin Developer version 6.1 program. The process used was to run the program using an initial `Level of Significance` of 0.1. If no link directed to the **result** node was formed the process was rerun doubling the `Level of Significance` until a network with at least one link directed to the **result** node was achieved. Since, in this problem all of the nodes except the **result** node have their values specified any nodes in the network with no links directed to the **result** node were removed. The remaining network was used for the testing. The overall classification error of the various learnt networks for disjoint training and test data sets was 67.69% and 67.38% for the general and expert models, respectively.

4.3.1. Complete seasons

The learned network using the general data for 1995/1996 season is shown in Fig. 3. It is possibly significant that the two nodes with the greatest number of dependencies are **dozzell** and **wilson**. We know from our other analysis that these are two important players, but with the network as shown we are unable to usefully include them. A crucial feature of this network is the **result** node has no children and its only parent is the **team_ranking** node. Since, in this problem the data for all the nodes except **result** are specified, we can infer the outcome of the game simply by knowing the quality of the opposition, the other attributes become irrelevant if the **team ranking** is specified. See Section 6 for further comment on this issue. Using the quality of the opposing team it is possible to correctly predict the outcome of 21 of the 38 games for the 1995/1996 season. This amounts to a reduction in the classification error of 13.15%. Using the expert data for the 1995/1996 season the network obtained is that shown in Fig. 4. This network correctly predicted 23 of the 38 games for the season a reduction in error of 18.42%. The Hugin BN learnt networks for the general and expert models for the 1996/1997 season are identical, consisting of the **team_ranking** and **result** nodes. These particular networks were extracted using a `Level of Significance` of 0.1 for both models.

4.3.2. Separate training and test data – single season

It is interesting to note that for the general model the attributes chosen by the Hugin learner for the periods in 1995/1996 season are a subset of those chosen by the MC4 learner for the same periods. There is a less strong relationship for the general model between the chosen attributes of the Hugin and MC4 learners for the 1996/1997, but still a lot of shared attributes. This is reasonable given that both learners are presumably choosing attributes with a strong correlation with the result. For both seasons the intra-season average classification error using the general data increased by 14.81%. Using the expert data set the average intra-season classification error increased by 20.37% and 5.55% for the 1995/1996 and 1996/1997 seasons, respectively Fig. 5.

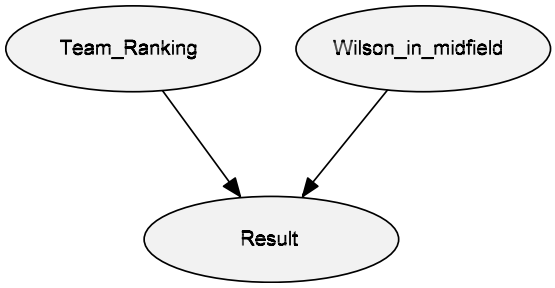


Fig. 4. Learnt BN for the expert model 95/96 season with `Level of Significance 0.1`.

4.3.3. Separate training and test data – cross seasons

Similar to the intra-season networks there is a striking similarity between the attributes chosen by the Hugin learner and the MC4 algorithm for the general model. We encountered a problem with the network produced by the Hugin learner for the period 2 and 3 general model data in the 1995/1996 season. This network crashed when we tried to run it so no results could be obtained for this training

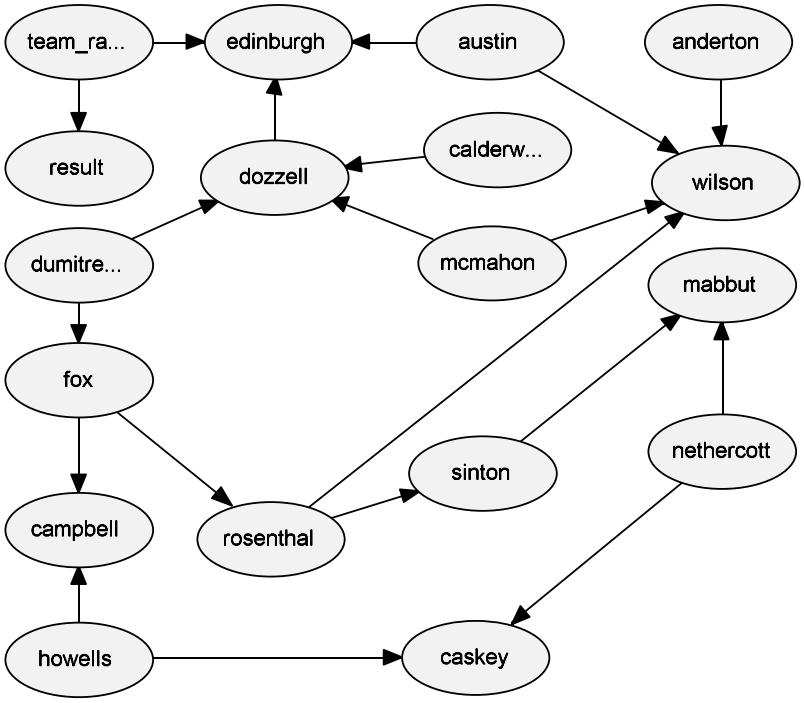


Fig. 3. Learnt BN for the general model 95/96 season with `Level of Significance 0.1`.

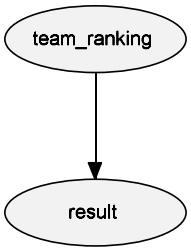


Fig. 5. Learnt BN for the general model 96/97 season with a `Level of Significance 0.1`.

period. The classification error for the cross season data showed reductions of 11.11% and 8.33% for the general and expert data sets, respectively.

4.4. K-nearest neighbour

The IB classifier from the MLC++ library is a version of the *K*-nearest neighbour algorithm. In effect the KNN algorithm constructs a graph with as many dimensions as we have attributes. We are not aware of an easy to interpret representation for graphs of high dimension so we provide no visual representation of the model constructed by this learner. We chose to use 3 neighbours for the voting comparison in this paper. Overall for the disjoint training and test data sets KNN proved to be an average performer with a classification error of 65.02% and 62.94% for the general and expert models, respectively. However, as expected with the same training and test data provided KNN performs exceptionally.

4.4.1. Complete seasons

For the 1995/1996 season KNN correctly predicts the result of 37 of the 38 games for the general model and 27 games for the expert model data. This amounts to an error reductions of 55.26% and 28.94%. For the 1996/1997 season the KNN algorithm again correctly predicts the result of 37 of the 38 games for the general model and 32 for the expert model giving error reductions of 50.00% and 36.84%, respectively.

4.4.2. Separate training and data – single season

With separate training and test data sets the performance of the KNN learner dropped dramatically, and interestingly providing more training data did not seem to improve its performance. The overall classification error for the 1995/1996 season for both general and expert models was 61.11% and for the 1996/1997 season it was 68.52% and 66.67% for the general and expert models, respectively.

4.4.3. Separate training and data – cross seasons

Cross season performance was generally a bit weak for the KNN learner. This might be because of an inability to filter out unimportant attributes involved in cross season changes. KNN produced an overall classification error for the cross season test periods of 65.44% for the general and 61.03% for the expert models respectively.

4.5. Validation and overfitting

In this problem we would not expect to get a completely accurate classification for the outcome of a given game. We have only a small sample of data a situation that will tend to cause a strong bias towards the specific data set. However, what we are interested here is in the relative performance of each learner and, since each learner could be expected to generate the same data set bias, the comparisons should be valid. We also have a situation in which the underlying mechanisms that determine the performance of the football team, the members of the team, their playing positions, fitness and tactics can all change. We would not expect our chosen attributes to account for all of the likely variations so its difficult to determine what is a reasonable level of predictive accuracy to expect.

4.6. Expert constructed Bayesian network

We already noted that the expert BN (Fig. 1) contained 3 nodes **Attack**, **Spurs_Quality**, and **Performance**, which do not directly represent any of the supplied attributes or the result. These nodes are a result of the model the expert has built to capture more detailed relationships between the attributes and the result than those provided by the other learners. Another difference with the expert BN is that is does not use the supplied training data for any of the tests. The structure of the BN and the value of the NPTs have all been fixed by the expert. This means it is unable to take into account any change that may occur outside of the expert chosen attributes. Despite these limitations, and the inherent analysis bias against the BN already discussed, the expert BN was the most accurate predictor of the outcome of the Spurs games with a classification error over the disjoint training and test data sets of 40.79%. Since, the expert BN only used the expert data set only one set of accuracy figures are given.

4.6.1. Complete seasons

The expert BN is the only learner we would not expect to appear overly accurate when looking at a complete season's data for both training and testing as it does not use training data. The expert BN did better than the most common value predictions for both the 1995/1996 and 1996/1997 seasons with a classification error of 40.79%.

4.6.2. Separate training and test data – single season

The expert BN had its poorest performance on the data for the 1995/1996 season. This is not difficult to understand given that: Sherringham played in every match for Spurs during that season; Anderton played only 6 matches in the season; Armstrong played in all bar one game of the season; Wilson only played in midfield in 3 games in the season. Thus given its chosen set of attributes there was little

variation the expert BN could produce over the 1995/1996 season. However, it is worth noting that with classification errors of 50.00% and 40.74% for the 1995/1996 and 1996/1997 seasons, respectively, it was still the best classifier for the intra-season data.

4.6.3. Separate training and test data – cross seasons

The expert BN produced the best results of any of the classifiers for every one of the cross season test periods. Since, it does not use the training data, any changes that occur between season not involving its key attributes are ignored. This is really a case of the expert being able to select the key features, and thus remove any other features which could adversely effect its predictions. However, in the case of something like a football team where over the course of a few seasons all the players may change it does potentially limit the useful lifetime of any given expert constructed BN. The classification error averaged 33.62% for the cross season data.

5. Predictive accuracy

Tables 1 and 2 show the relative accuracy of the different learners in predicting the outcome of the games using the general and expert model data, respectively. When using the same training and test data for the complete seasons all of the learners perform significantly better than the most common assumption with KNN as the best performer. When disjoint training and test data sets were used the performance of the KNN learner dropped significantly and the expert BN outperformed all the other learners. The learners generally performed similarly with both the general and expert chosen data sets.

6. Conclusions and way forward

The process of machine learning, and learning in general, provides us with two tangible benefits, understanding and prediction. While it is true that the better our understanding the better we should be able to make predictions, it is possible to make accurate predictions with limited understanding. We can treat these as qualitative and quantitative results from the learning process. The understanding we gain from the learning process allows us to construct models which reflect what we have learned about the relationships between the attributes and the relative importance of each attribute. In terms of the football matches it lets us see which of the selected attributes are the crucial factors effecting the outcome of a game, and gives some clues as to the relationships between some of those factors.

The different learning techniques vary in what they provide in terms of understanding of the interrelationships between the attributes and the outcome of a game. The MC4 learner identifies those attributes which have the largest effect on the outcome of the game. It shows their relationships to each other in terms of their effect on the outcome of the game. This is a very simplified model of the game itself. The naive Bayesian learner does not construct a model as such, its model is predefined. The learning process for the naive Bayesian learner is then simply one of discovering the relative strength, and polarity, of the effect of each attribute with respect to the result. The learnt BN looks for correlations between the values of the attributes including the result. Once a BN is constructed using the correlations that lie within the required sensitivity, then the NPTs can be learnt from the available data. KNN does not construct a model as such, it simply uses the existing data and provides a *likeness* comparison with any test data. Thus KNN does not significantly enhance our understanding. The expert constructed BN represents the knowledge of the expert, that is, it is a model is the expert's belief of the interrelationships between the attributes and their relative importance. One of the limitations of all the non expert methods used here is that they only use the supplied attributes. This is particularly limiting in its effect on the learnt BNs. In a problem where most of the supplied attributes have defined values the possible network structures for a learnt BN are very restricted and, in effect, become just reduced versions of the naive Bayesian model. While they are not observed the nodes **Attack**, **Spurs_Quality**, and **Performance** in the expert BN help build a model of the games Spurs played. This model gives us some additional insight into how the observed attributes effect the outcome of the game.

Given the inherent analysis bias against the BN model, its performance is genuinely impressive. Although the model has now long been irrelevant (since it contains variables relating to key players who have retired or left the club) the results here tend to confirm the excellent potential of BNs when they are built by a reliable domain expert. The ability to provide accurate predictions without requiring much learning data are an obvious bonus in any domain where data are scarce. Moreover, the BN was relatively simple for the expert to build and its basic structure could be used again in this and similar types of problems.

There are a number of directions in which future work could be done. As pointed out this method of prediction is inherently asymmetric. It should be possible to construct a more symmetrical model using similar data for all the teams in the league. However, this would involve at least multiplying the amount of computational work by the number of additional teams in the league. Another obvious potential improvement would be to qualify the inherent quality of each player who plays – a simple 3-point scale based on objective criteria like international performances could be feasible. This approach would provide much greater *longevity* to the model. Also, learning from the expert BN here, we could use abstract nodes like 'attack quality' and 'defence quality' to both improve the model and ensure its longevity.

References

[1] J. Pearl, Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgan-Kaufmann, San Mateo, CA, 1988.
[2] N.E. Fenton, P. Krause, M. Neil, Software measurement: uncertainty and causal modelling, IEEE Software 10 (4) (2002) 116–122.

[3] N.E. Fenton, M. Neil, The jury observation fallacy and the use of Bayesian networks to present probabilistic legal arguments, Mathematics Today (Bulletin of the IMA) 36 (6) (2000) 180–187.
[4] A. Joseph, N.E. Fenton, M. Neil, Predicting football results using Bayesian nets and other machine learning techniques (version with full set of models and data), (2005), http://www.dcs.qmw.ac.uk/~norman/papers/Spurs-2.pdf.
[5] M. Neil, N. Fenton, S. Forey, R. Harris, Using Bayesian belief networks to predict the reliability of military vehicles, IEE Computing and Control Engineering Journal 12 (1) (2001) 11–20.
[6] D. Heckerman, D. Geiger, D. Chickering, Learning Bayesian networks: the combination of knowledge and statistical data, Machine Learning 20 (1995) 197–243.
[7] T. Cover, P. Heart, Nearest neighbour pattern classification, IEEE Transactions on Information Theory 13 (1967) 21–27.
[8] J.R. Quinlan, Induction of decision trees, Machine Learning 1 (1986) 81–106.
[9] Håvard Rue, Øyvind Salvesen, Prediction and retrospective analysis of soccer matches in a league, preprint Statistics 10 (1997).
[10] Tom M. Mitchell, Machine Learning, McGraw-Hill, New York, 1997.
[11] Ronny Kohavi, Dan Sommerfield, MLC++ machine learning library in C++, SGI http://www.sgi.com/tech/mlc/ (1996).
[12] Andrew Bradley, Brian Lovell, Michael Ray, Geoffrey Hawson, On the methodology for comparing learning algorithms: a case study, Australian and New Zealand Conference on Intelligent Information Systems, 37–41 (1994).

Appendix B

Learnt Graphs

This appendix has various graphs learnt by either the Bnlearn package learners or LUMIN during our analysis. The graphs can be instructive in understanding the strengths and weaknesses of each learner, and the text in chapter 6 refers to these graphs. The following graphs are from the ALARM network.

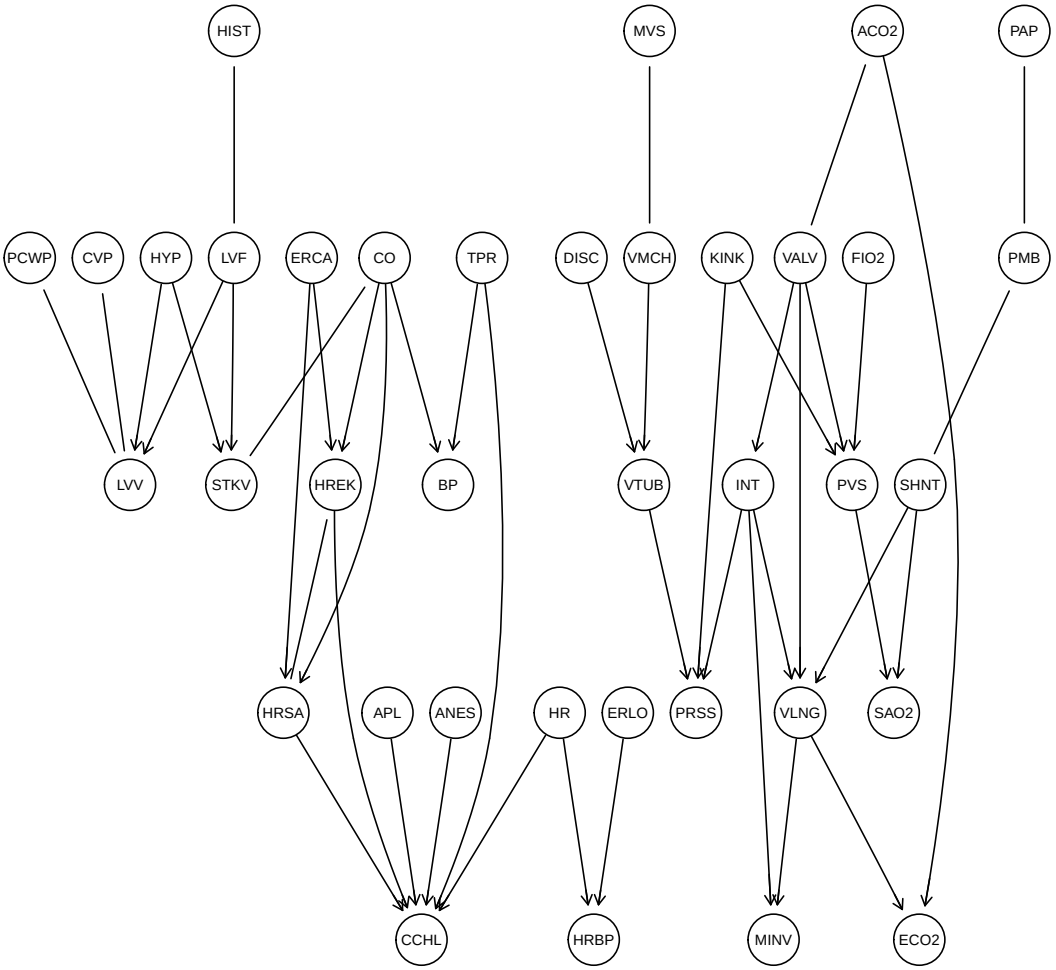


Figure B.1: ALARM Network Learnt by Grow-Shrink Learner - $\alpha = 0.1$

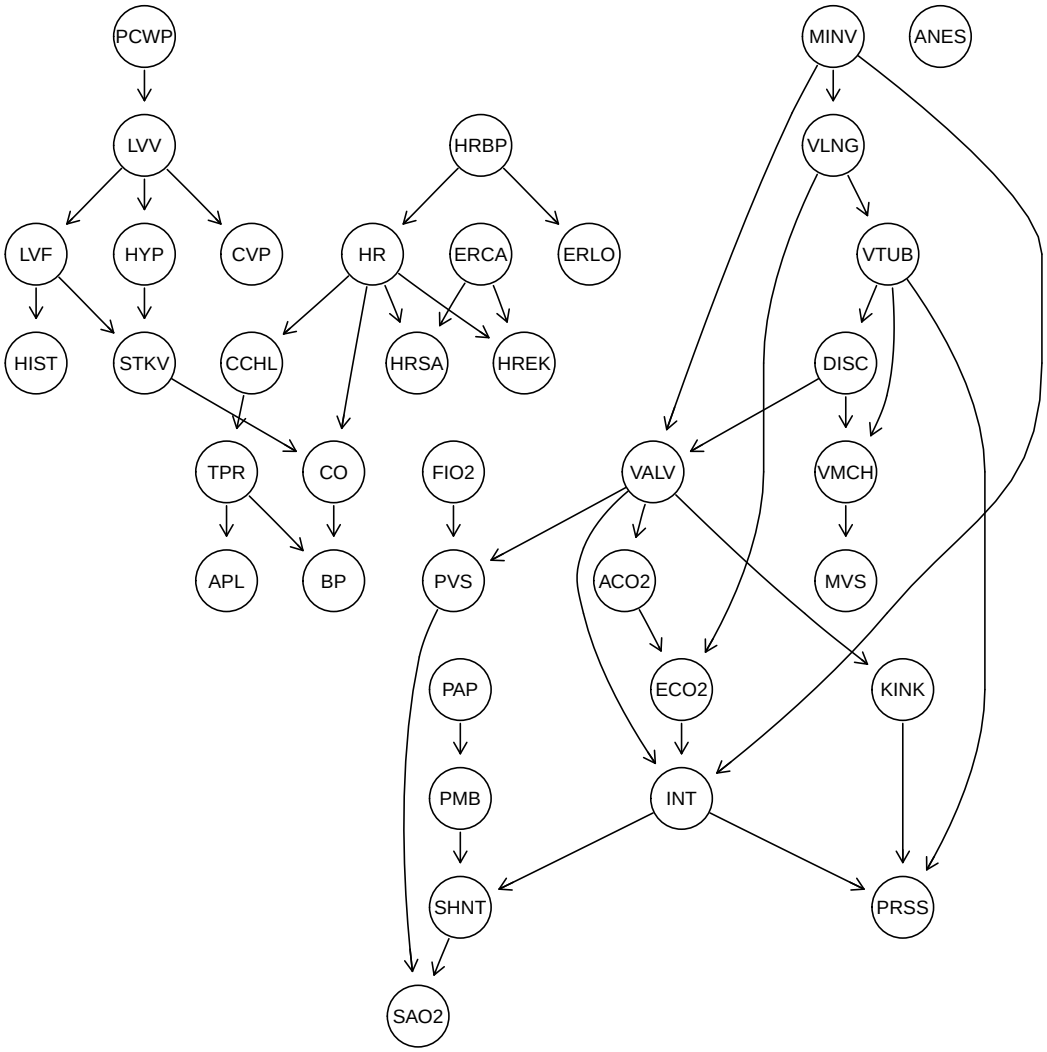


Figure B.2: ALARM Network Learnt by Max-Min Hill-Climbing Learner - $\alpha = 0.1$

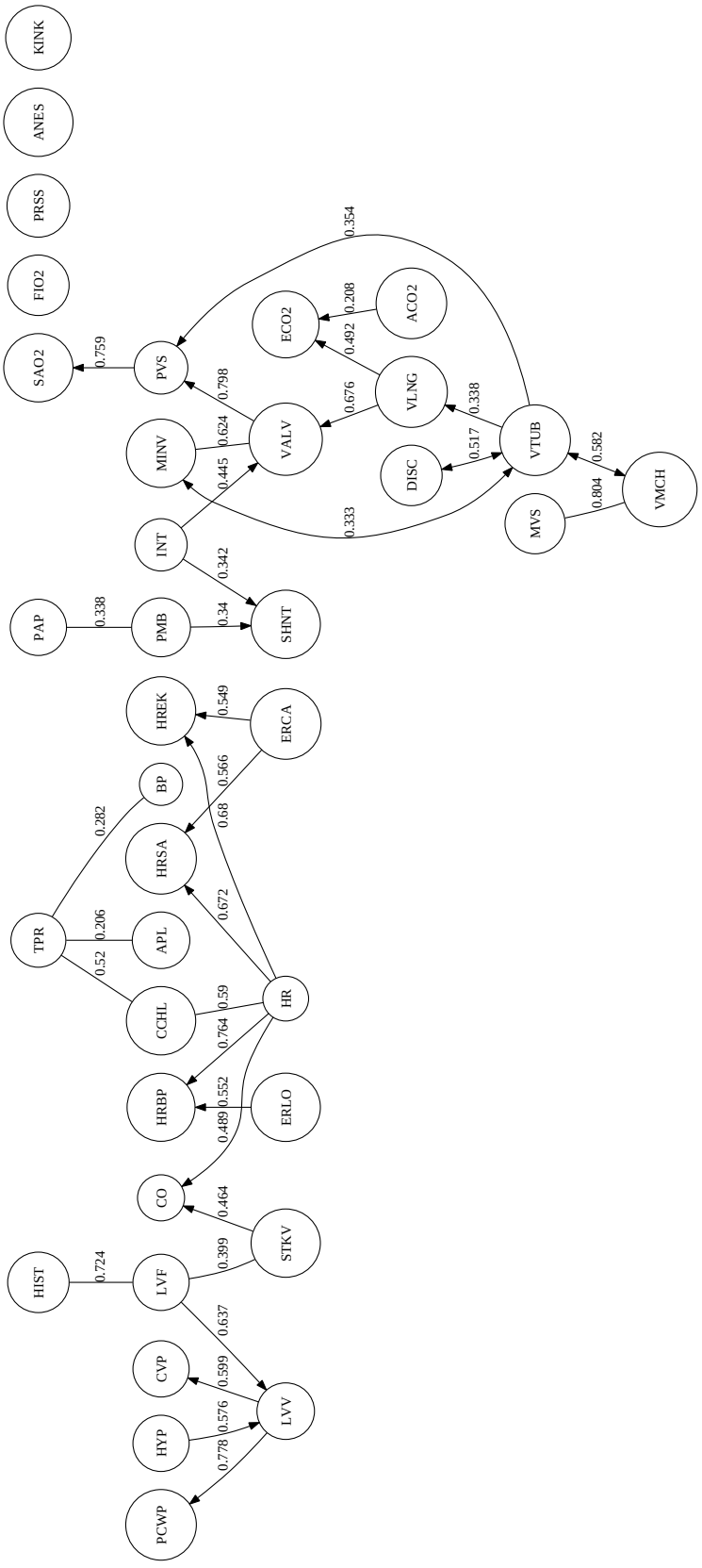


Figure B.3: ALARM Network Learnt by LUMIN with NMI 0.2 and NCMI $\delta = 0.7$ with Heuristic Link Removal

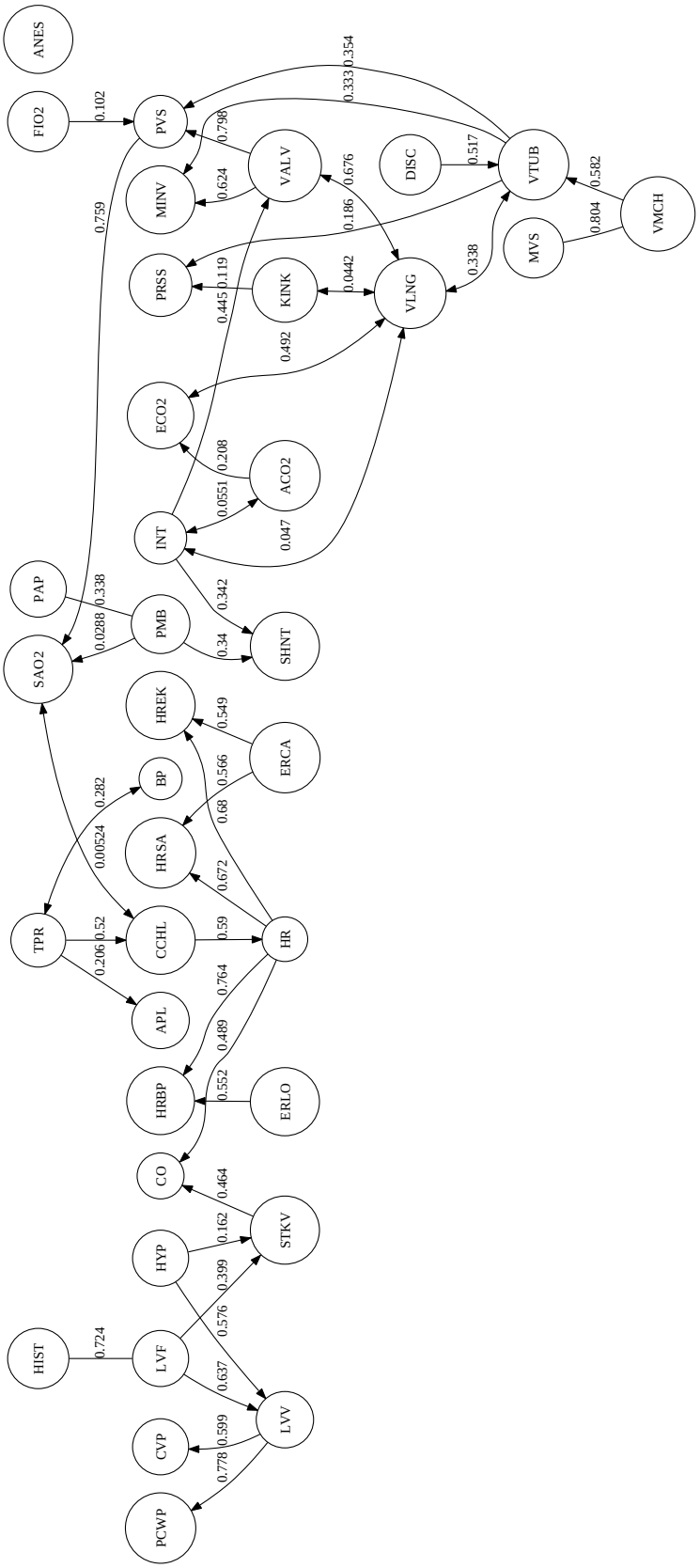


Figure B.4: ALARM Network Learnt by LUMIN with NMI 0.005 and NCMI 0.9 with Heuristic Link Removal

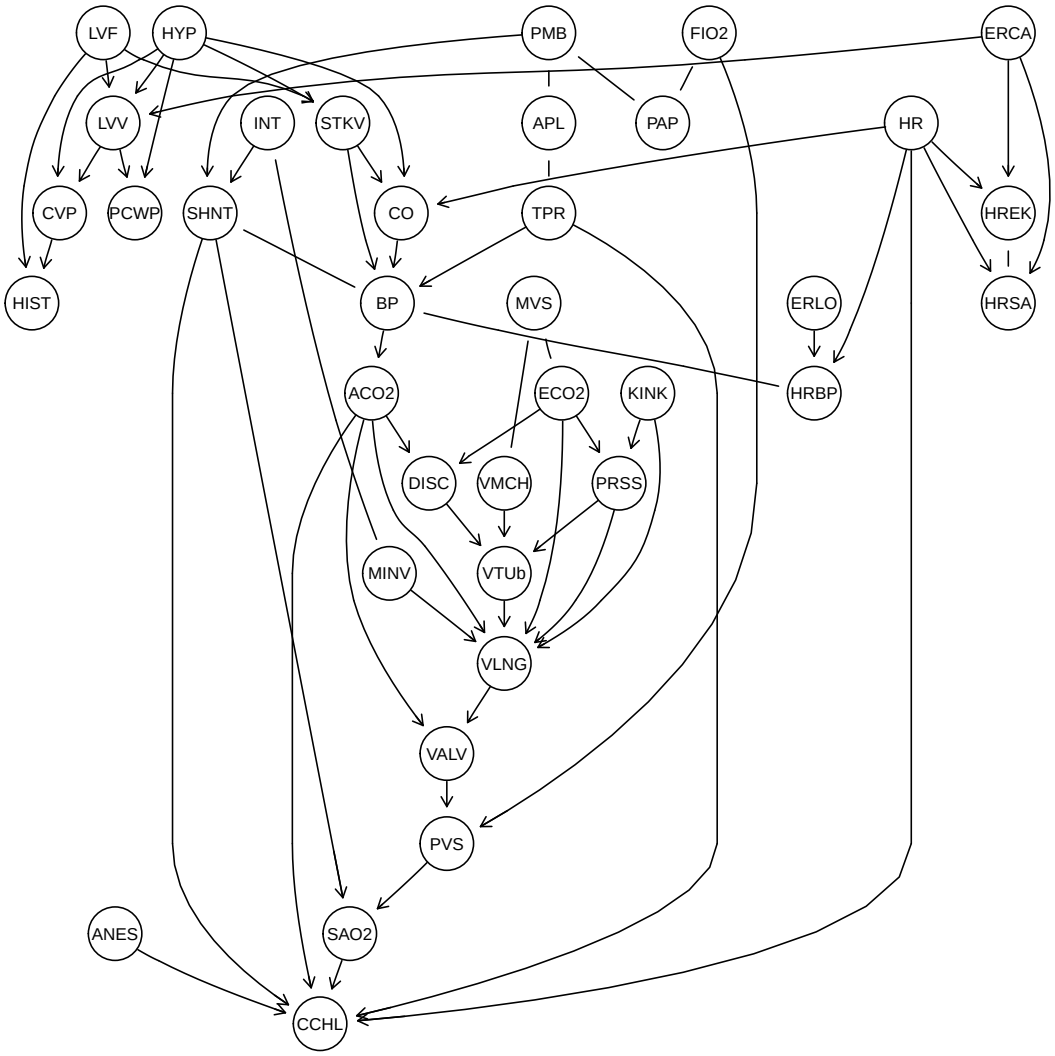


Figure B.5: ALARM Network Learnt by Grow-Shrink Learner - $\alpha = 0.1$

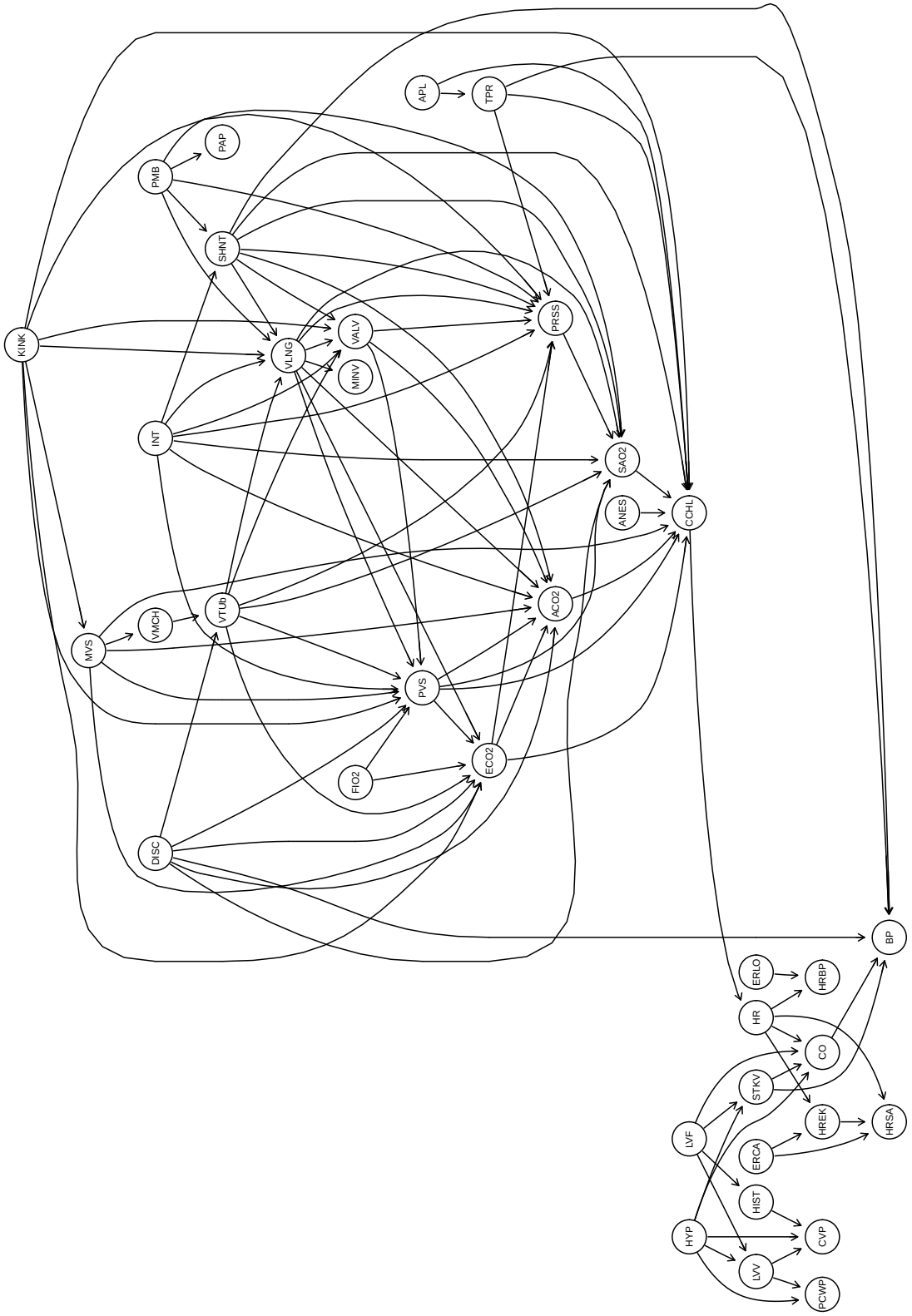


Figure B.6: ALARM Network Learnt by Hill Climbing Learner

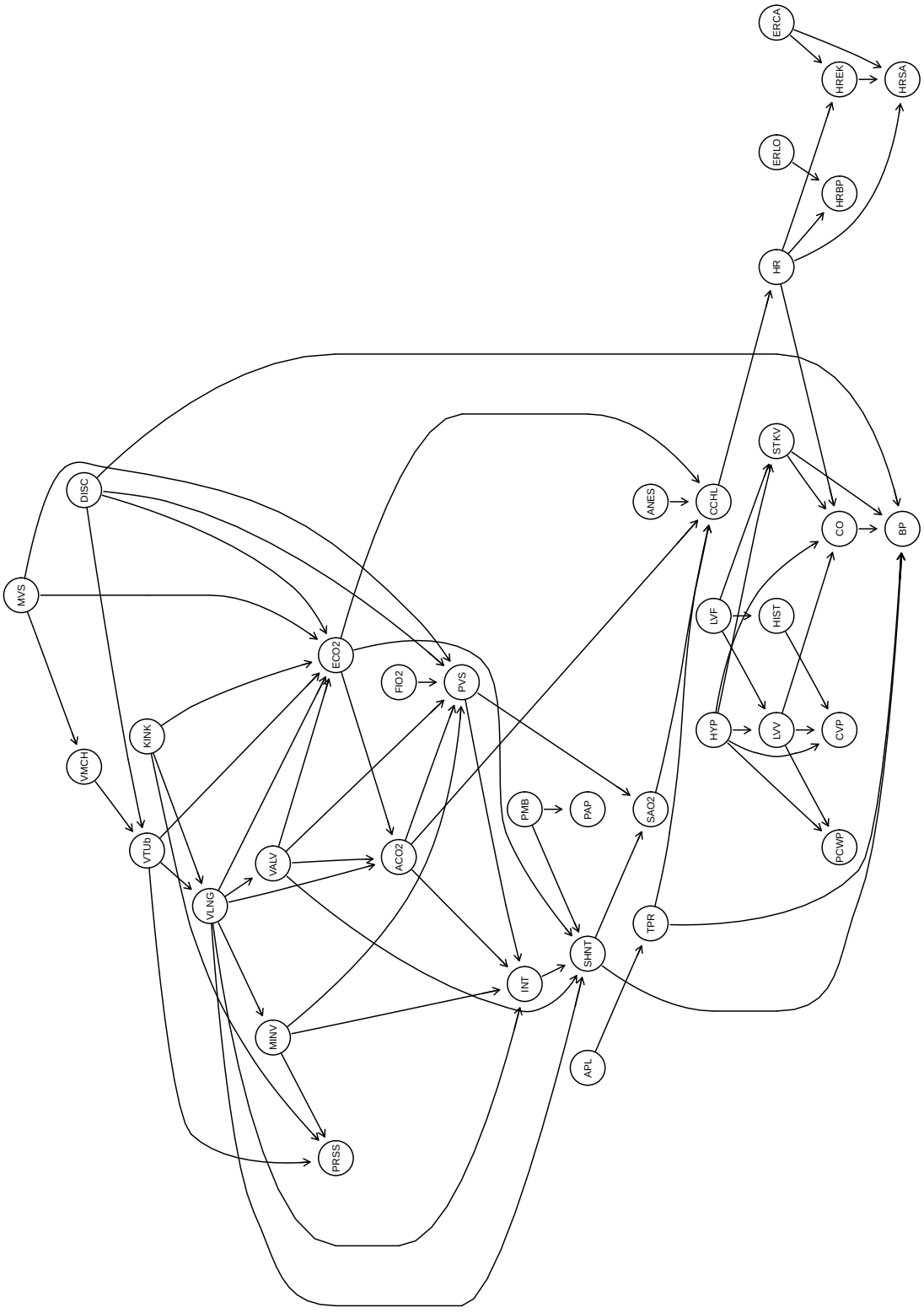


Figure B.7: ALARM Network Learnt by Max-Min Hill Climbing Learner - $\alpha = 0.1$

Figure B.8: ALARM Network Learnt by LUMIN with NMI 0.2 and NCMI 0.5 with no Heuristic Link Removal

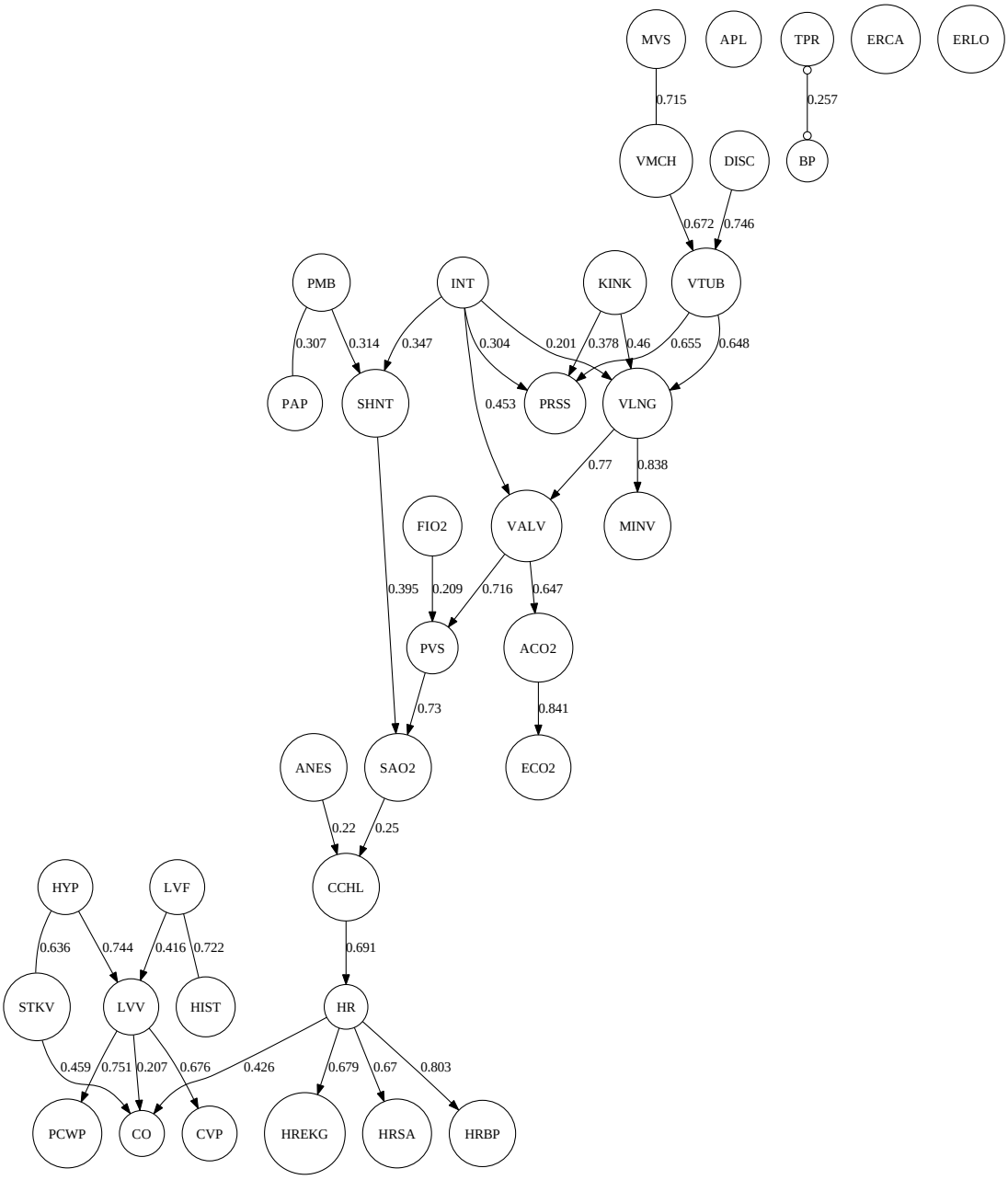


Figure B.9: ALARM Network Learnt by LUMIN with NMI 0.2 and NCMI 0.7 with Heuristic Link Removal

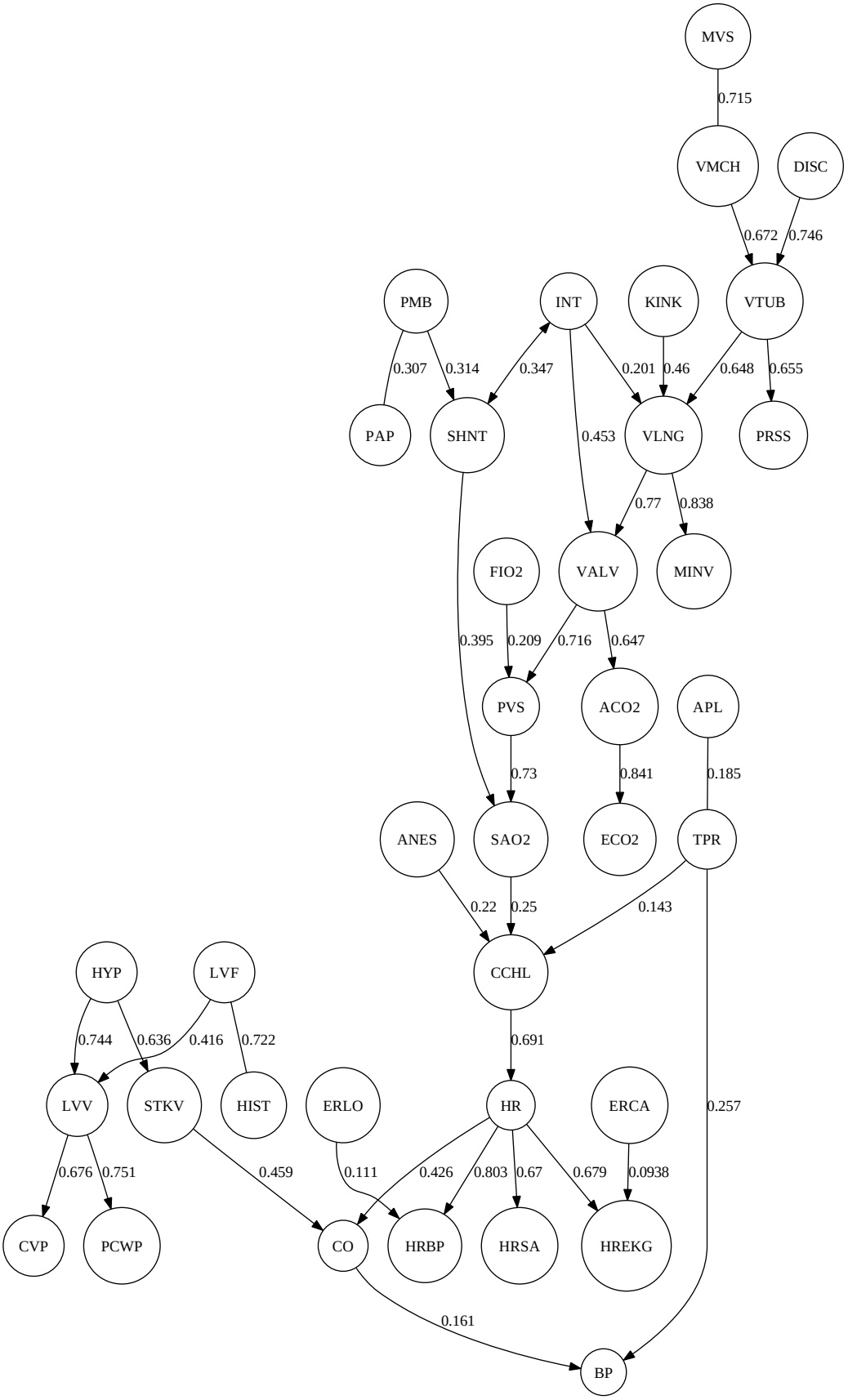


Figure B.10: ALARM Network Learnt by LUMIN with NMI 0.05 and NCMI 0.9 with Heuristic Link Removal

The following graphs are from the Hailfinder network.

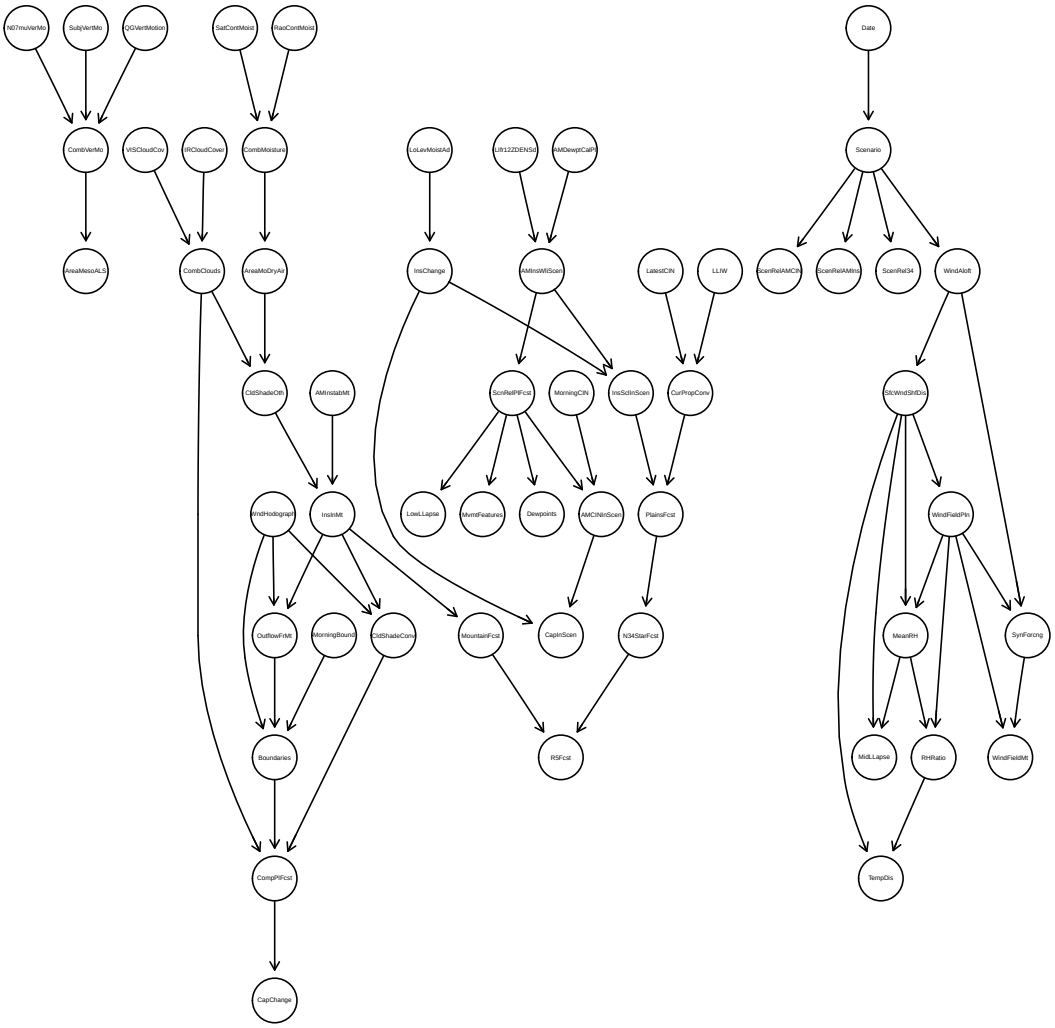


Figure B.11: Hailfinder Network Learnt by the Max-Min Hill Climbing Learner with $\alpha = 0.1$

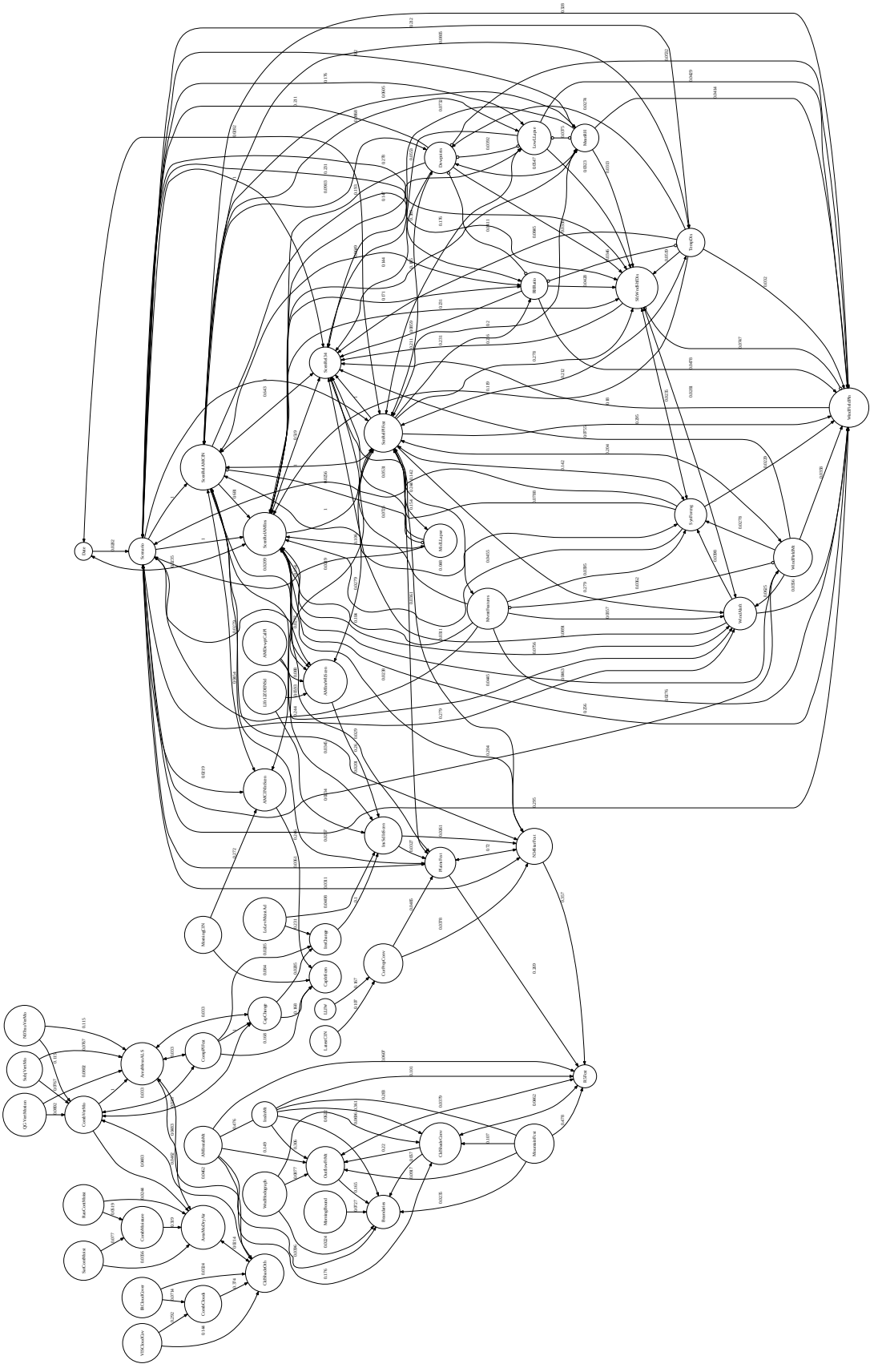


Figure B.12: Hailfinder Network Learnt by LUMIN with NMI 0.02 and NCMI 0.5 without Heuristic Link Removal

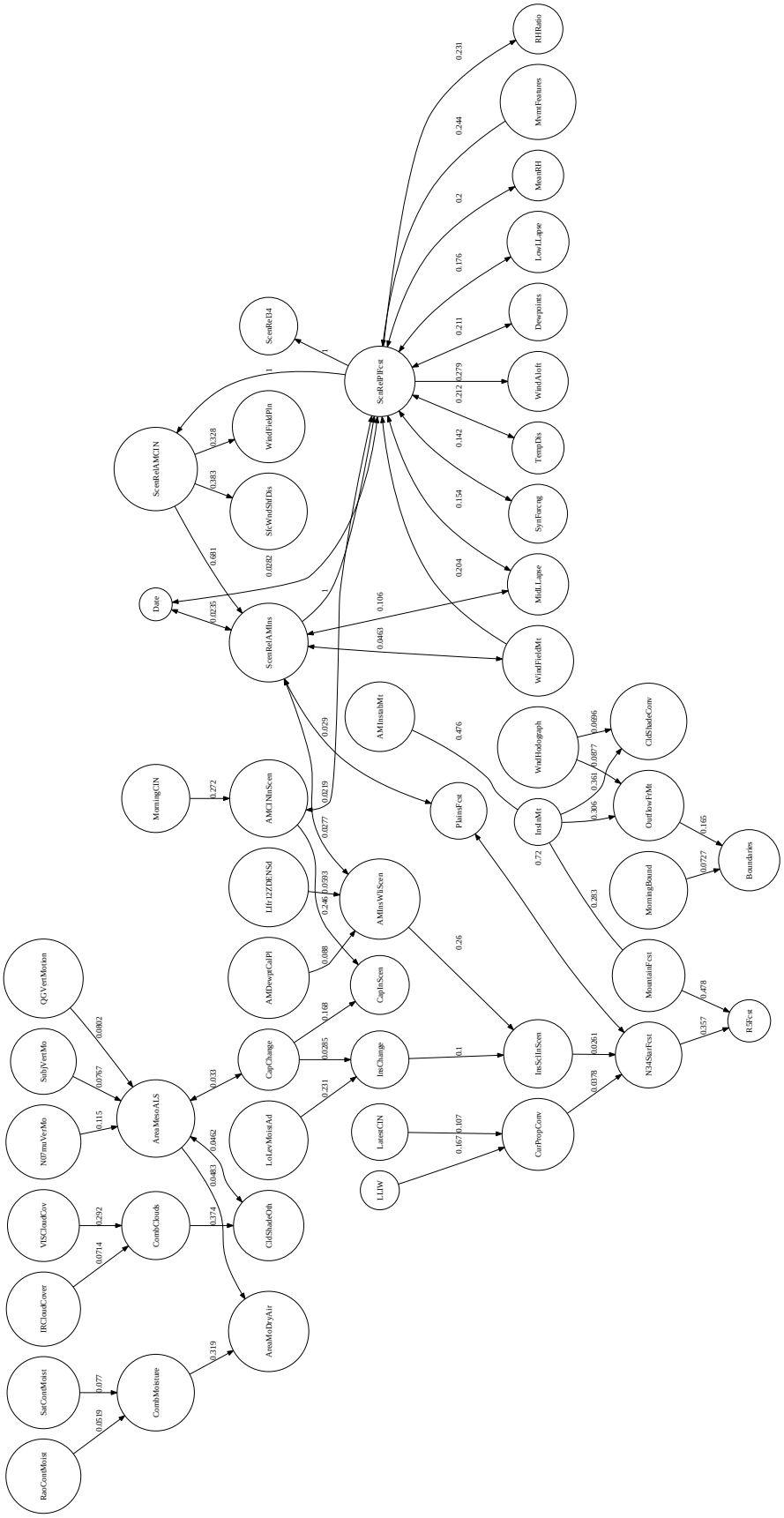


Figure B.13: The Hailfinder Network Learnt by LUMIN with NMI of 0.02 and NCMI 0.5 with heuristic link and similar node removal

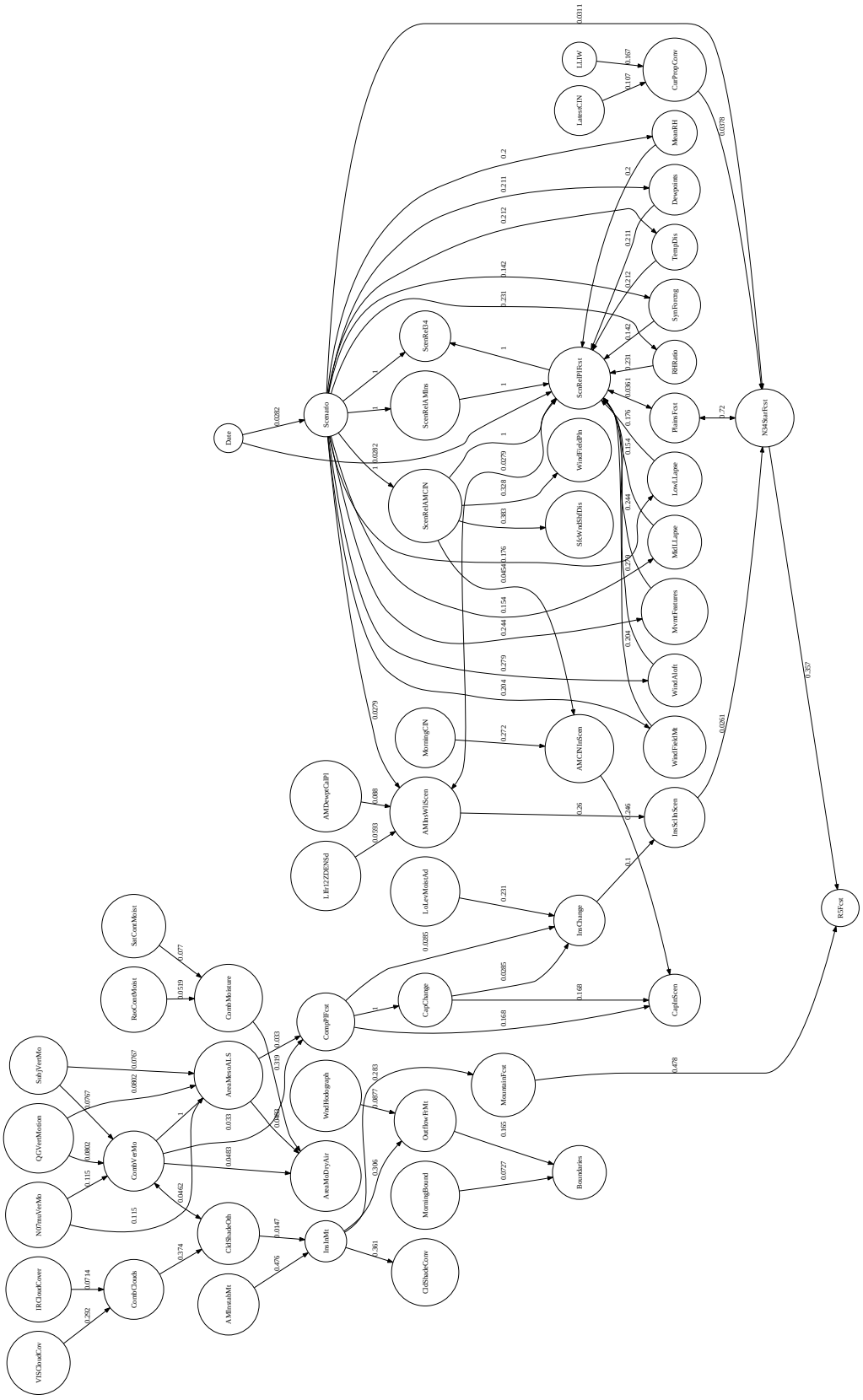


Figure B.14: The Hailfinder Network Learnt by LUMIN with NMI of 0.01 and NCMI 0.9 with Heuristic Link Removal and Domain Information

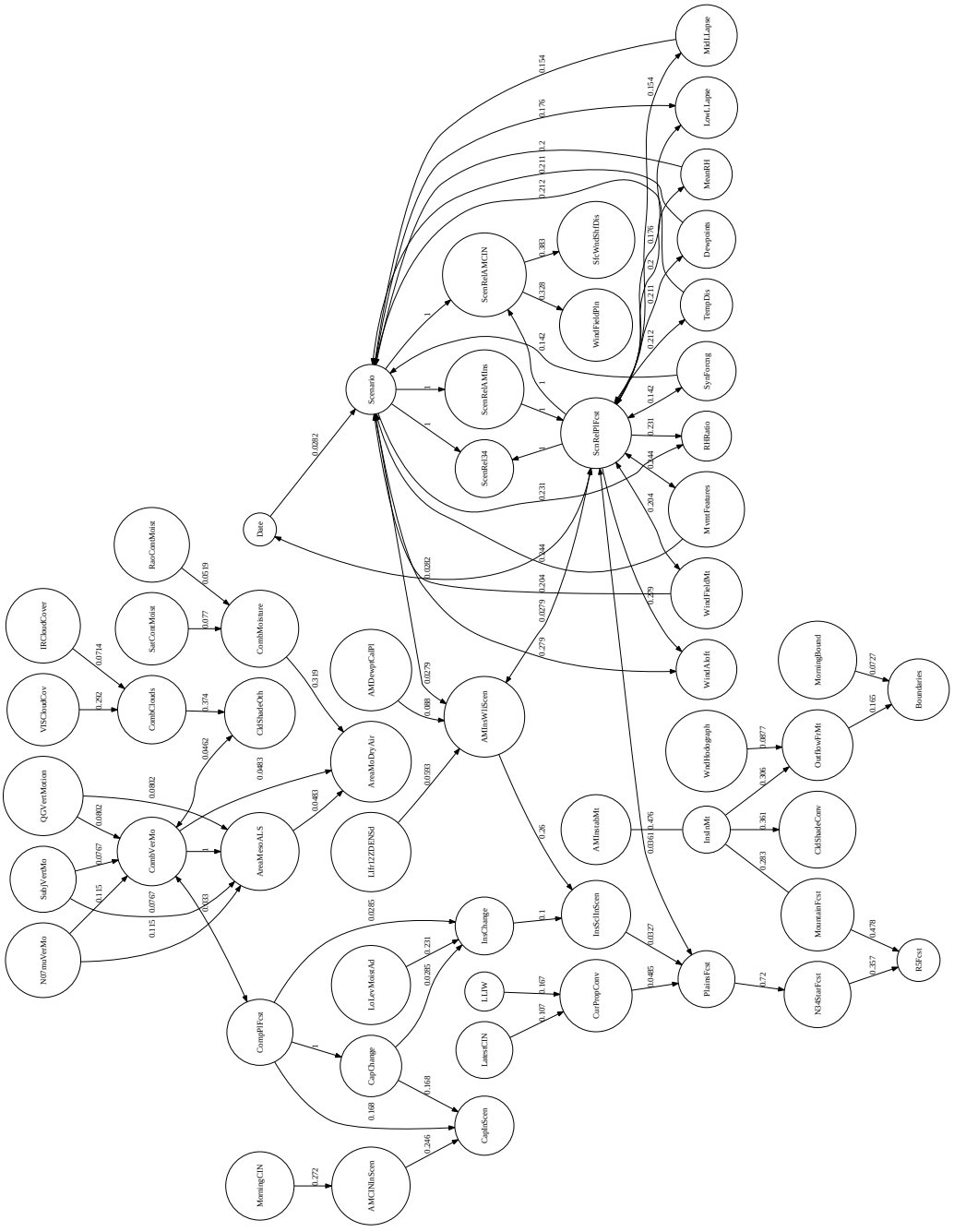


Figure B.15: The Hailfinder Network Learnt by LUMIN with NMI of 0.02 and NCMI 0.9 with Heuristic Link Removal

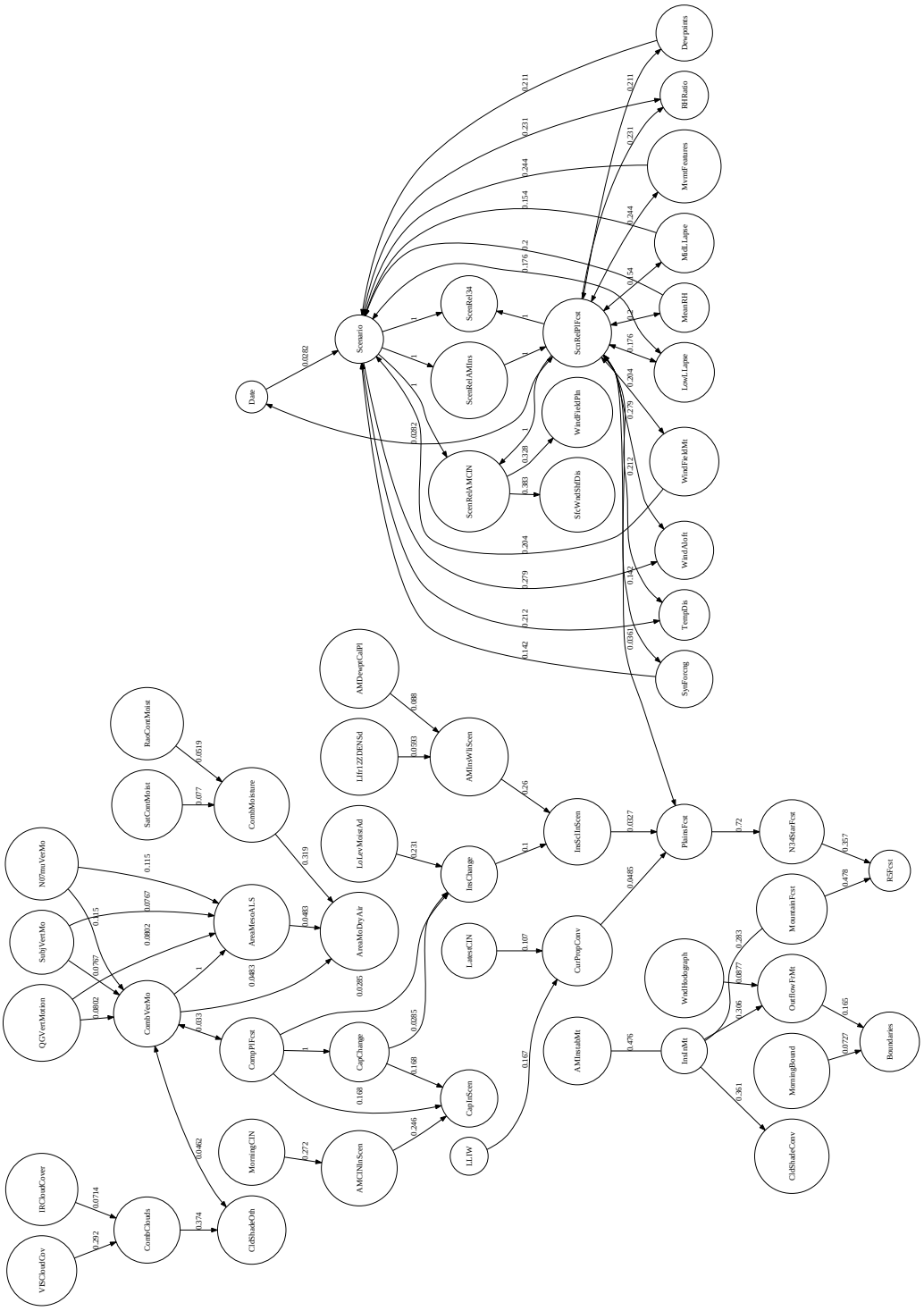


Figure B.16: The Hailfinder Network Learnt by LUMIN with NMI of 0.02 and NCMI 0.7 with Heuristic Link Removal

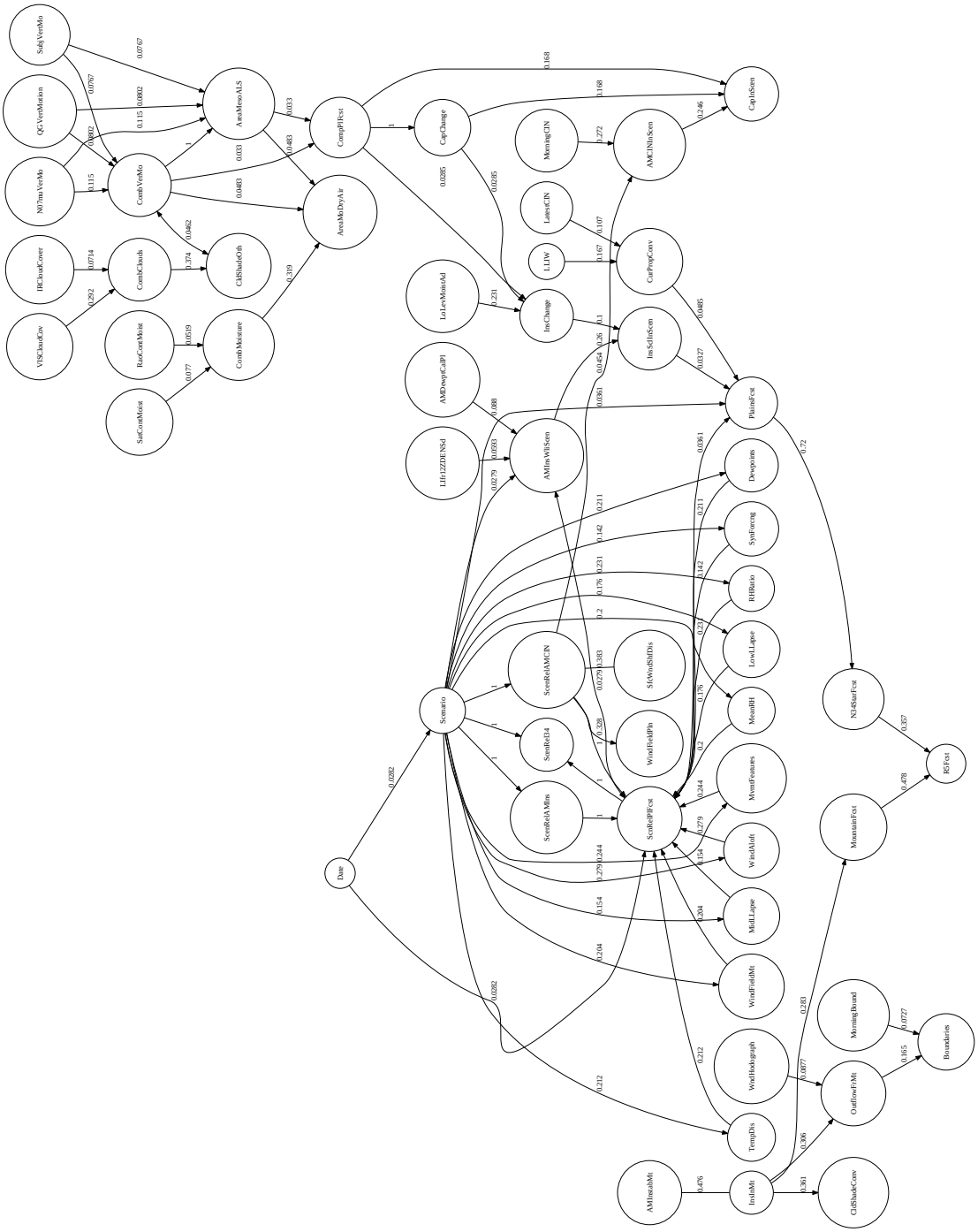


Figure B.17: The Hailfinder Network Learnt by LUMIN with NMI of 0.02 and NCMI 0.7 with Heuristic Link Removal and Domain Information

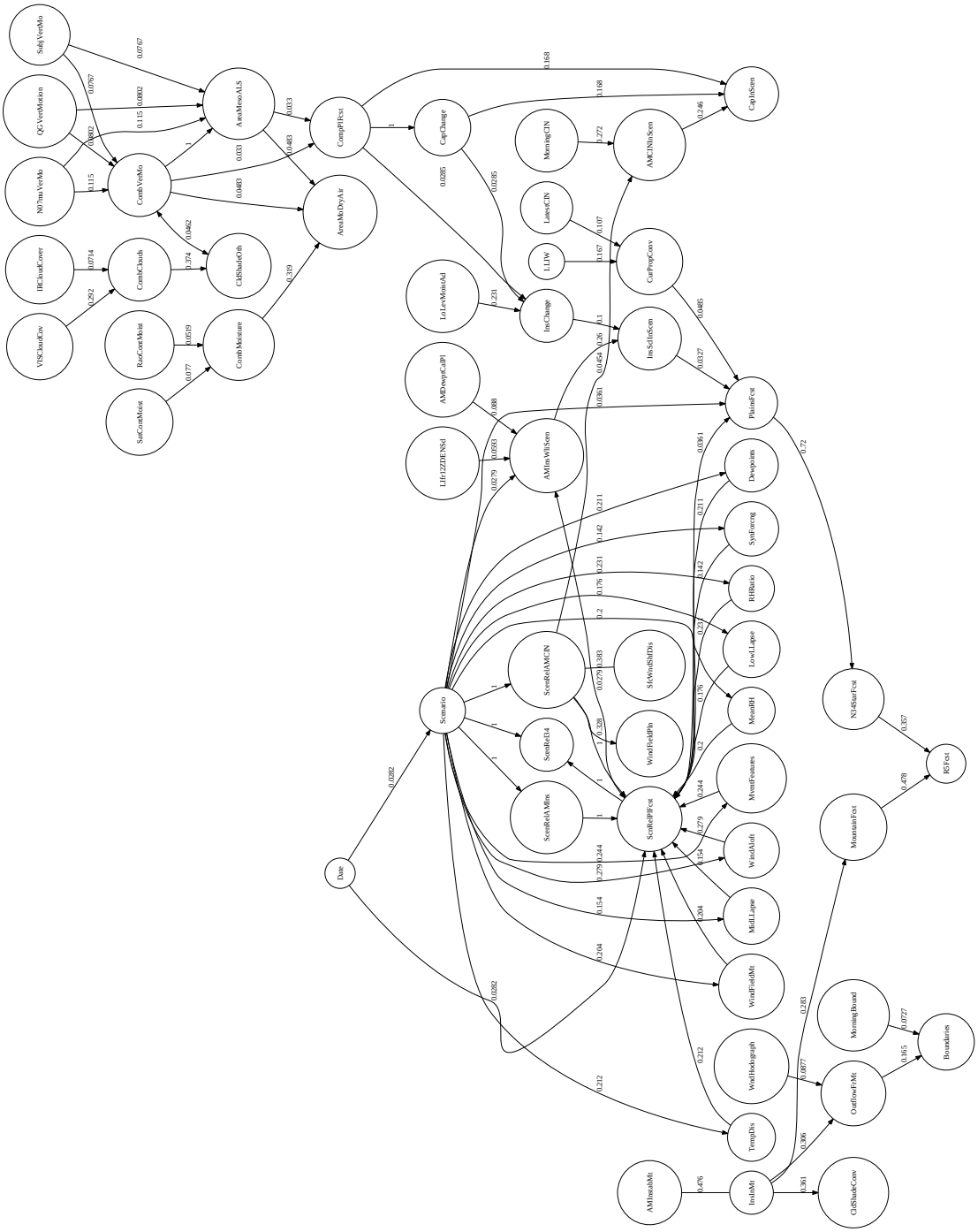


Figure B.18: The Hailfinder Network Learnt by LUMIN with NMI of 0.02 and NCMI 0.9 with Heuristic Link Removal and Domain Information

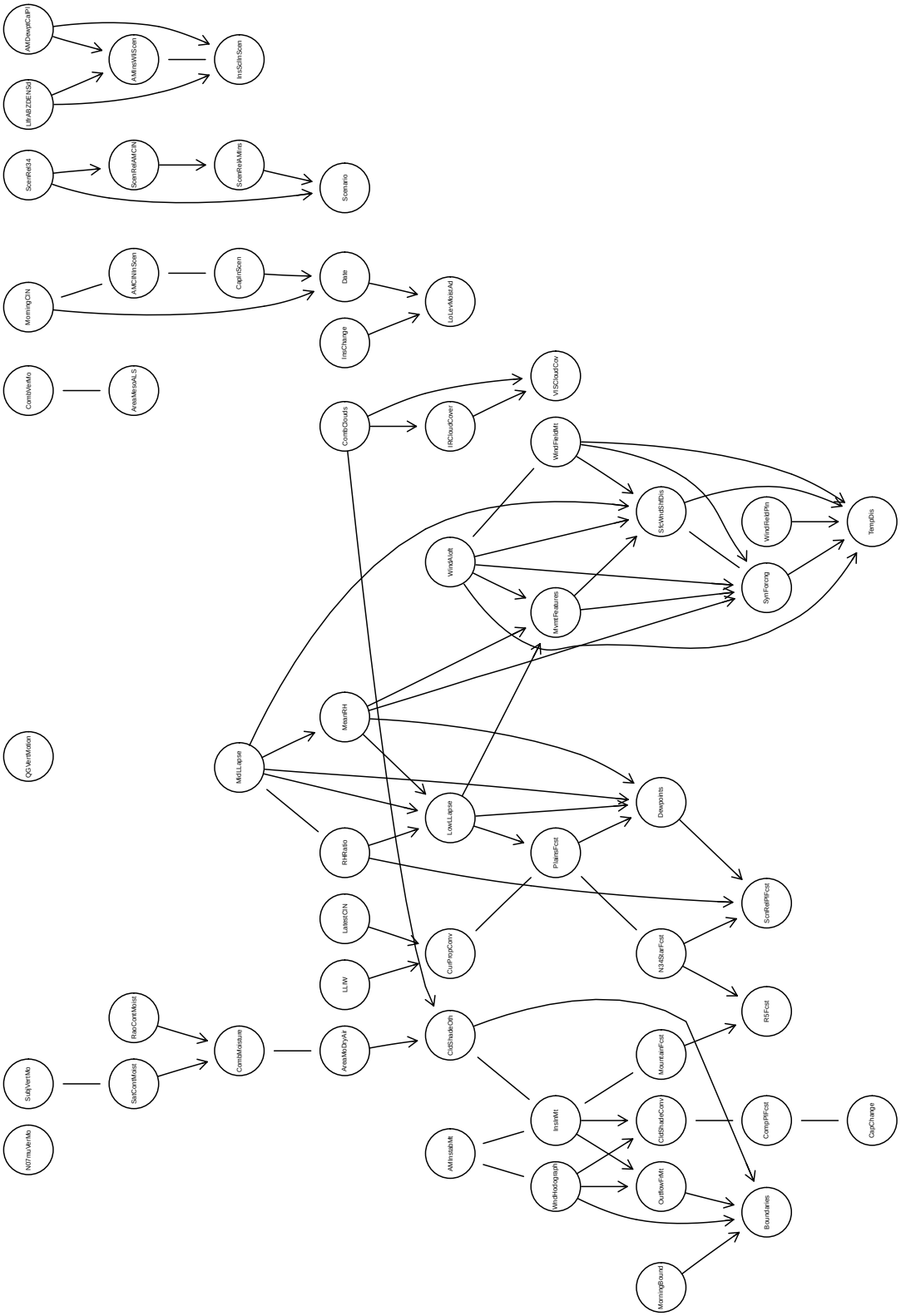


Figure B.19: Hailfinder Network Learnt by the Grow-Shrink Learner with $\alpha = 0.1$

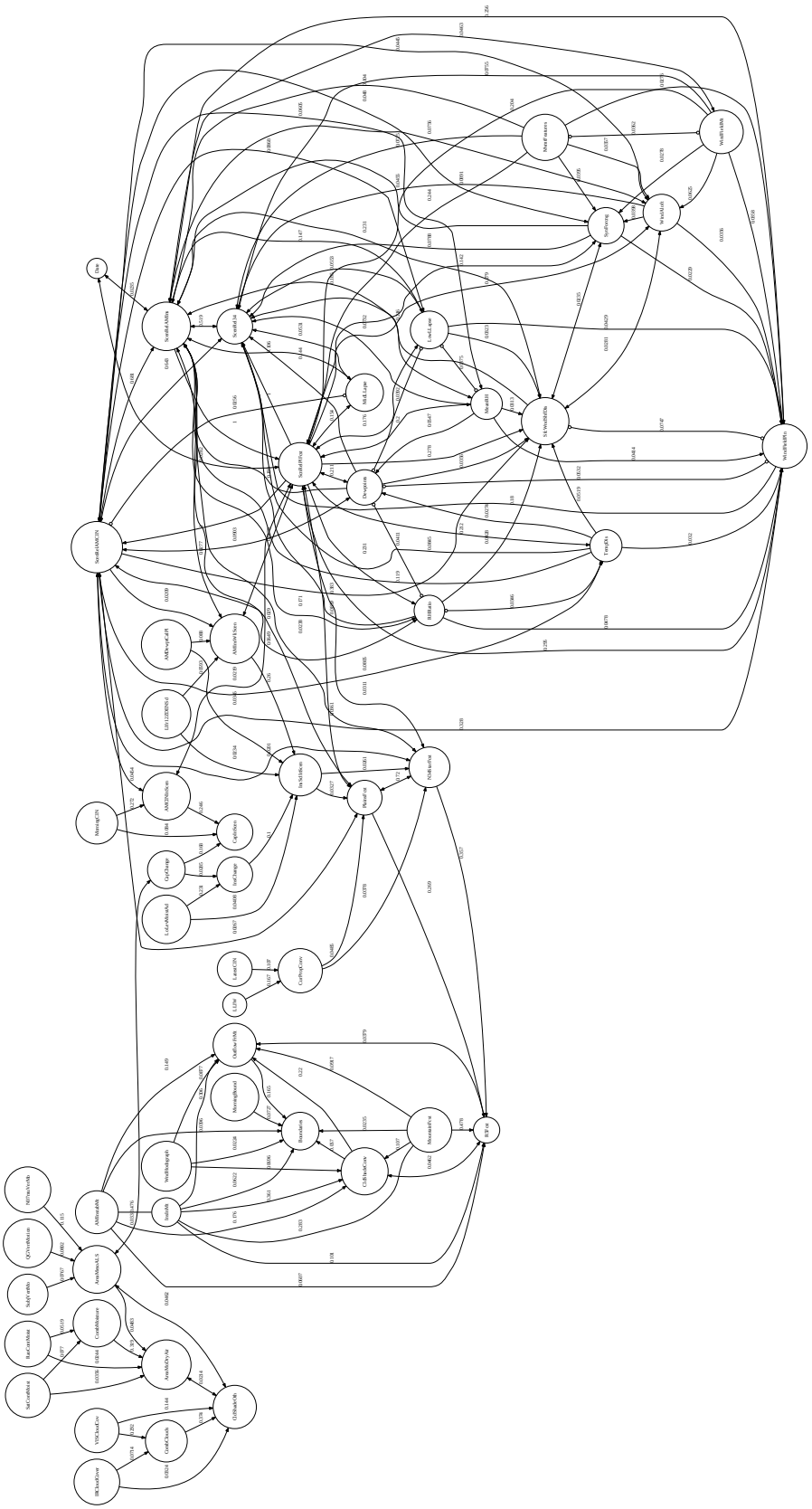


Figure B.20: The Hailfinder Network Learnt by the LUMIN Learner with NMI of 0.02 and NCMI of 0.5 with similar node removal

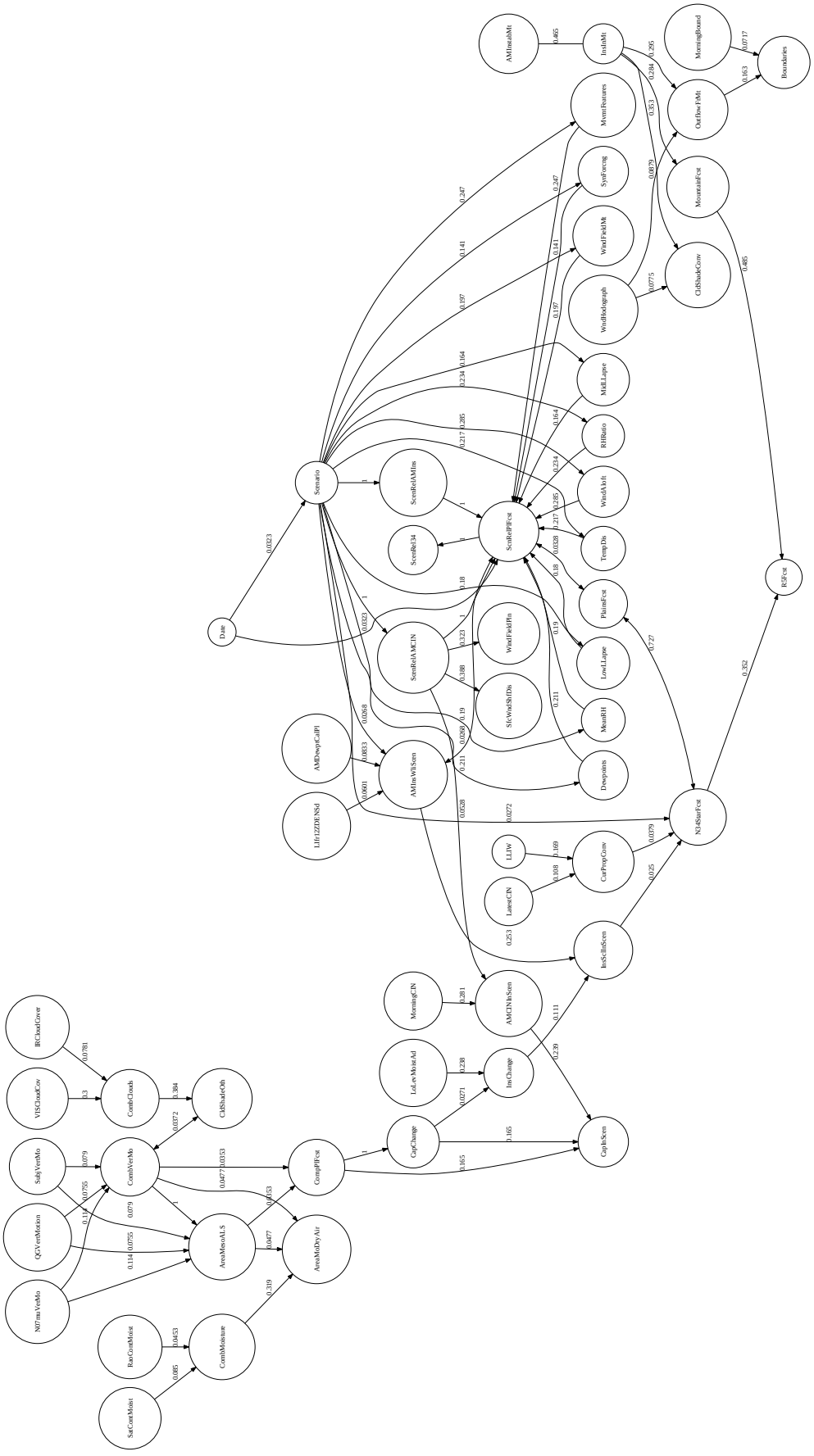


Figure B.21: Hailfinder Network Learnt by the LUMIN Learner with $NMI = 0.02$, $NCMI = 0.5$ and using some Domain Information.

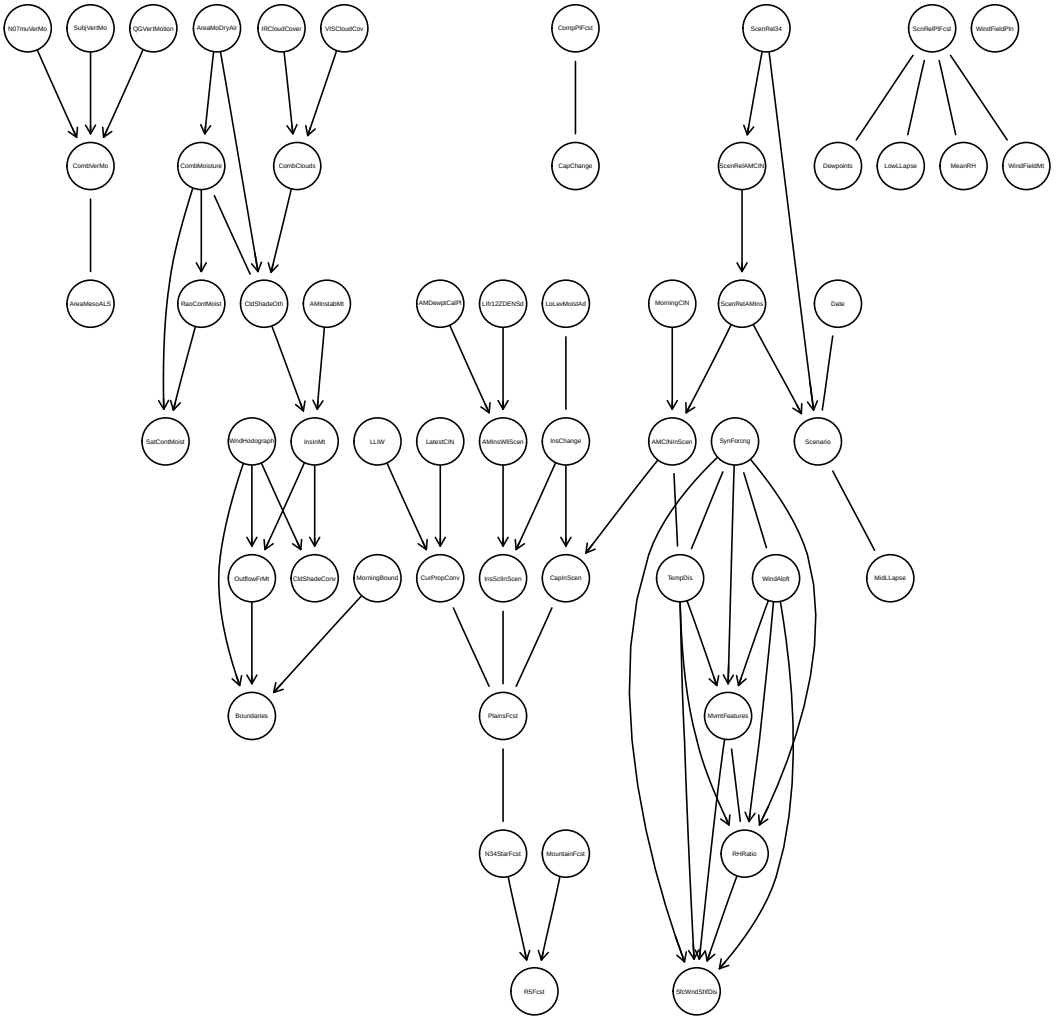


Figure B.22: Hailfinder Network Learnt by the Grow-Shrink Learner with $\alpha = 0.1$

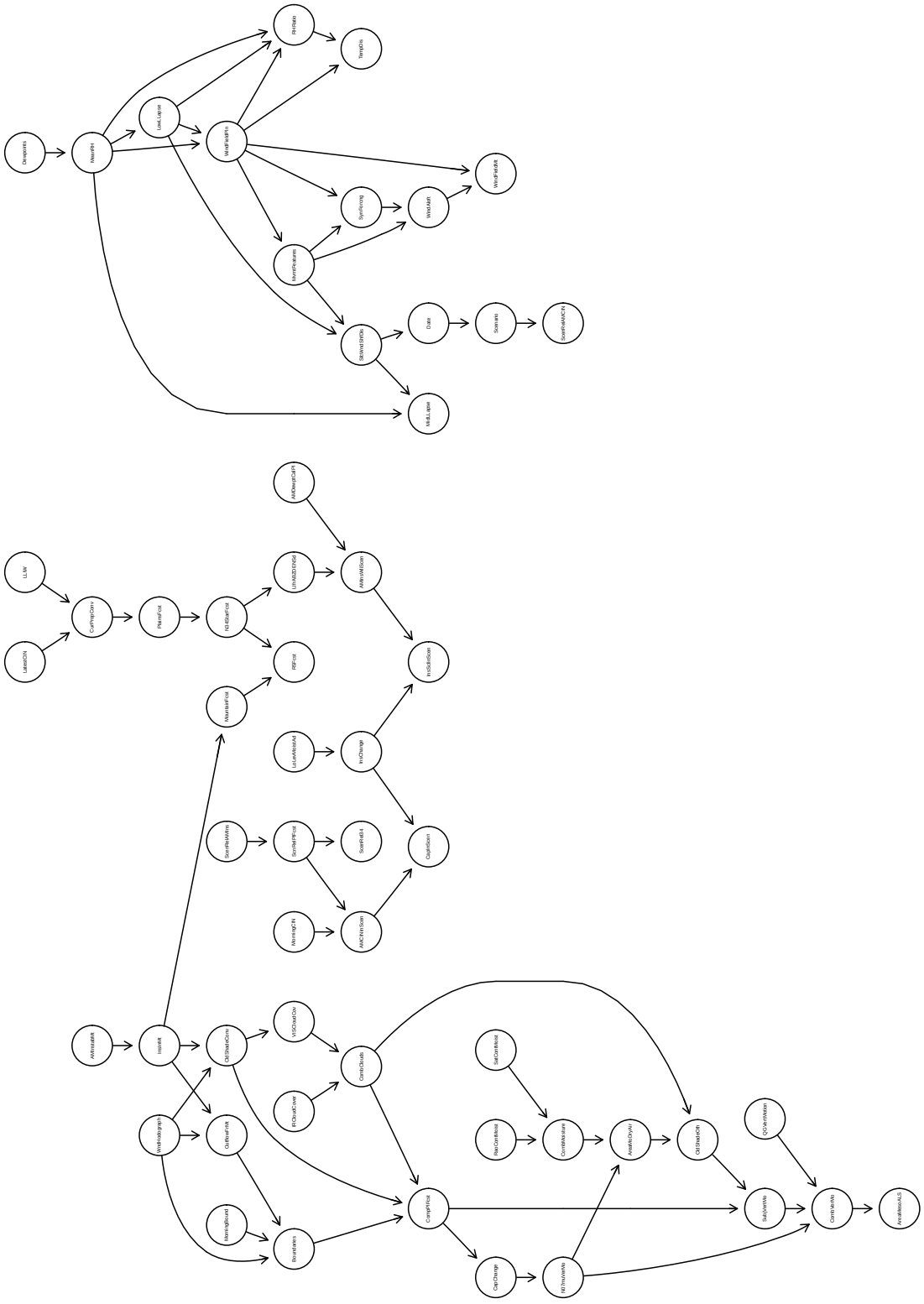


Figure B.23: Hailfinder Network Learnt by the Max-Min Hill Climbing Learner with $\alpha = 0.1$

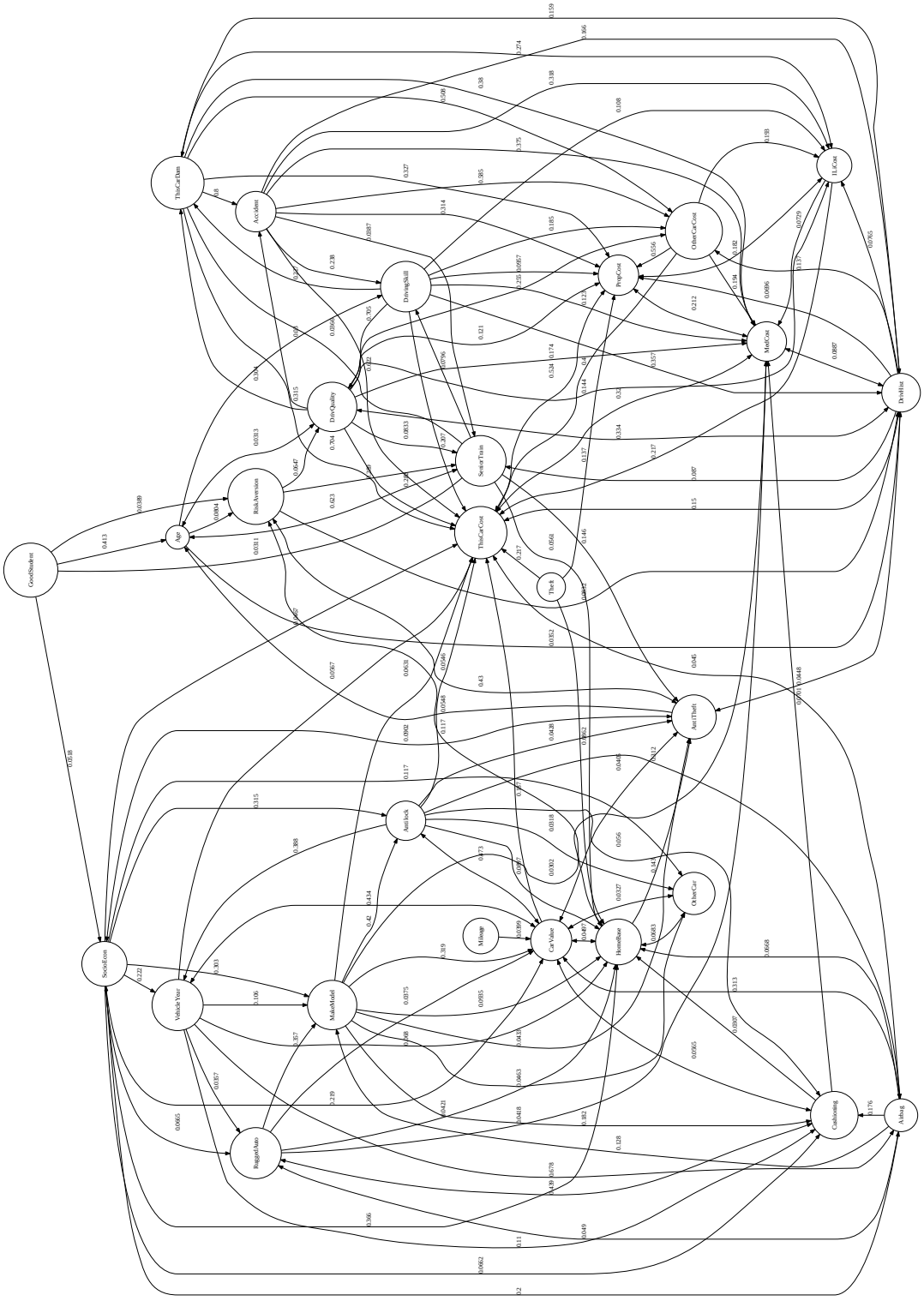


Figure B.24: Insurance Network Learnt by the LUMIN Learner with $NMI = 0.03$ and $NCMI = 0.5$ with no Heuristic Link Removal

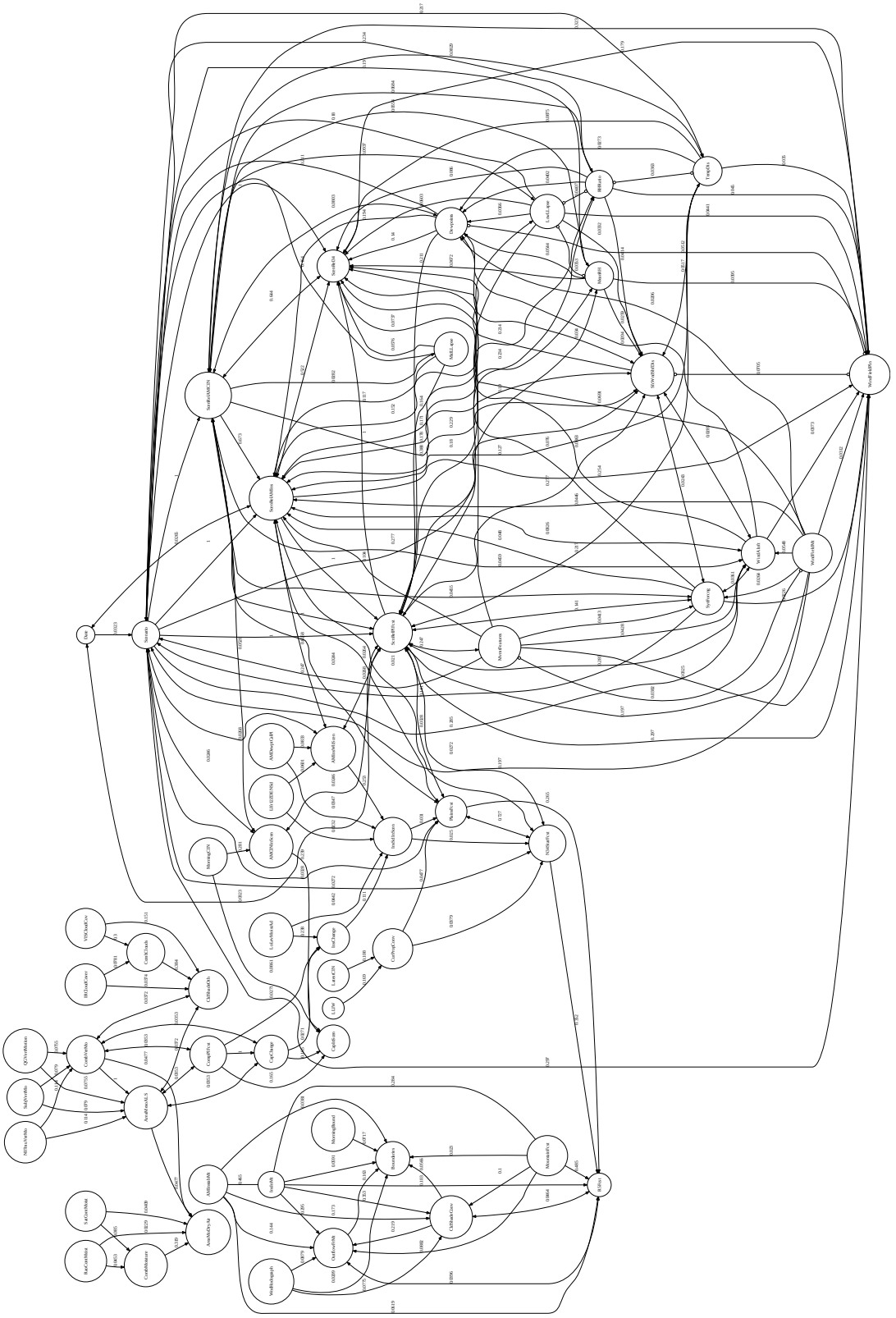


Figure B.25: Hailfinder Network Learnt by LUMIN with NMI 0.2 and NCMI 0.5 without Heuristic Link Removal

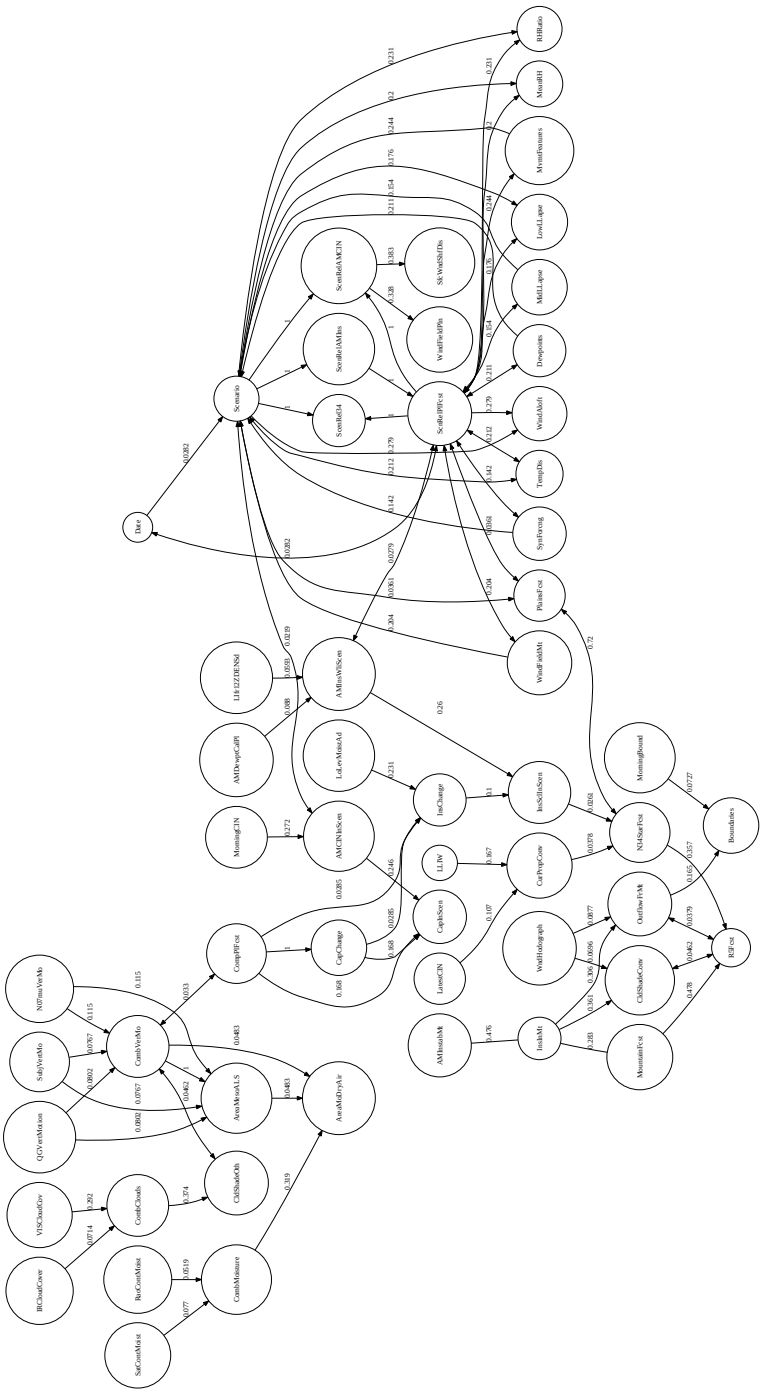


Figure B.26: The Hailfinder Network Learnt by LUMIN with NMI of 0.02 and NCMI 0.5 with Heuristic Link Removal

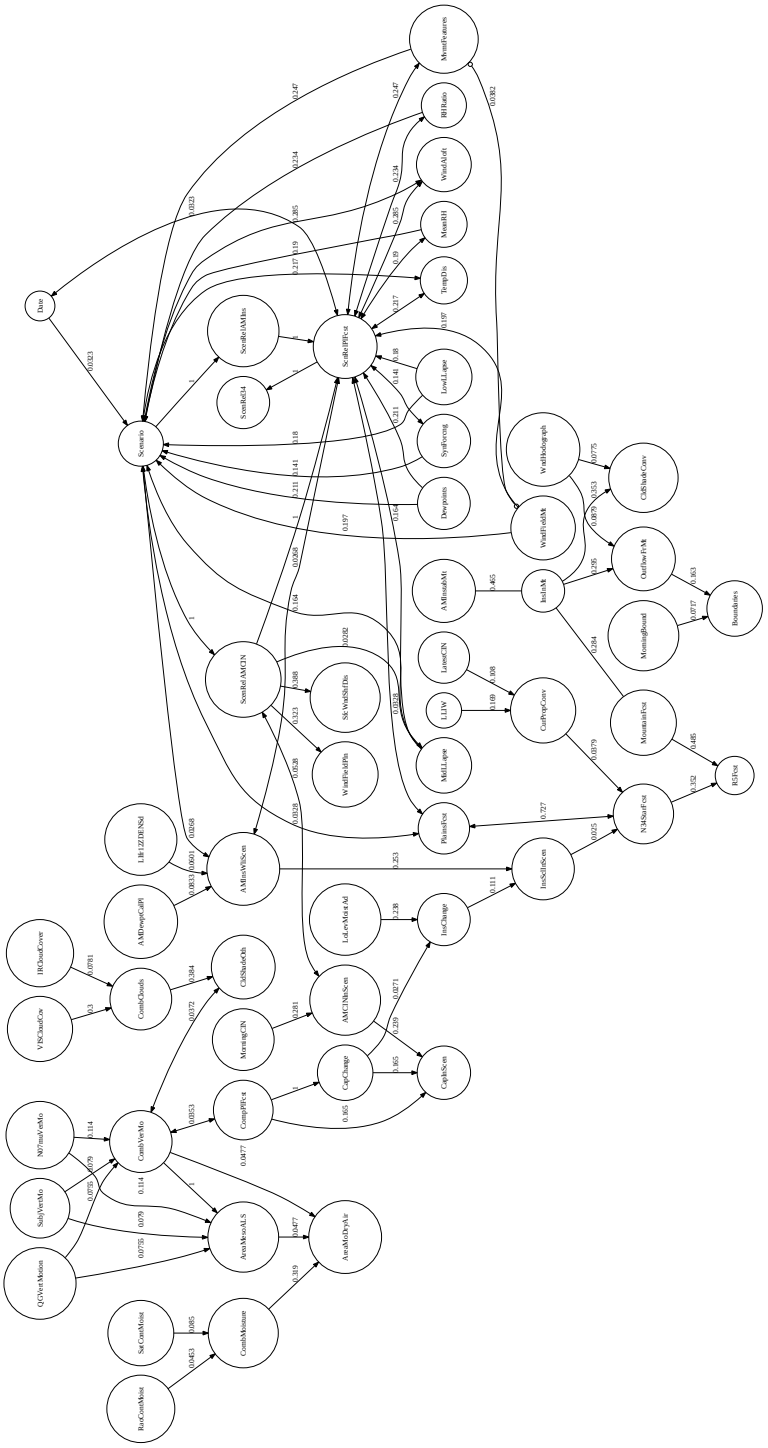


Figure B.27: Hailfinder Network Learnt by the LUMIN Learner with $NMI = 0.02$, $NCMI = 0.5$ with Heuristic Link Removal

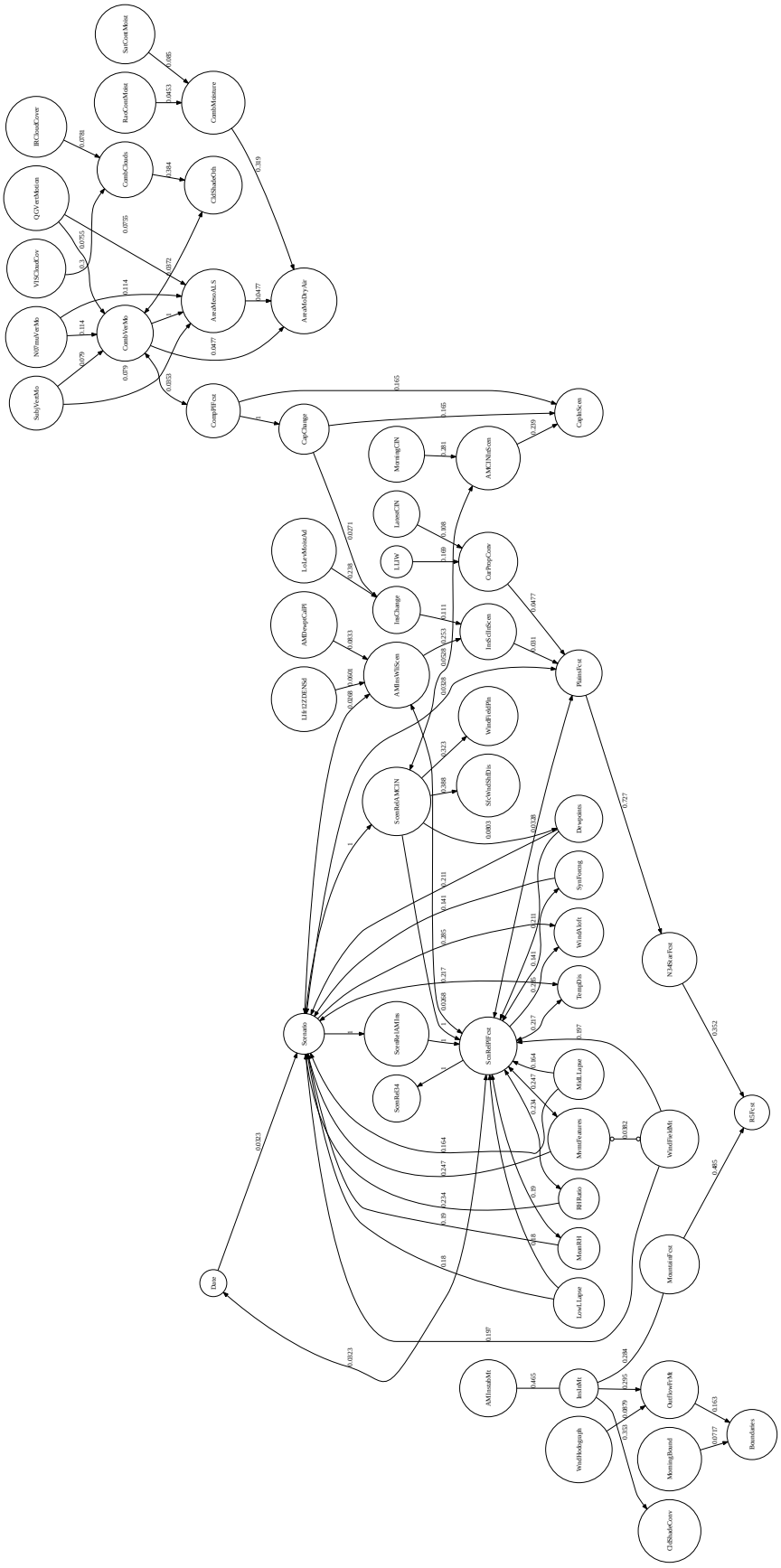


Figure B.28: Hailfinder Network Learnt by the LUMIN Learner with $NMI = 0.02$, $NCMI = 0.7$ with Heuristic Link Removal

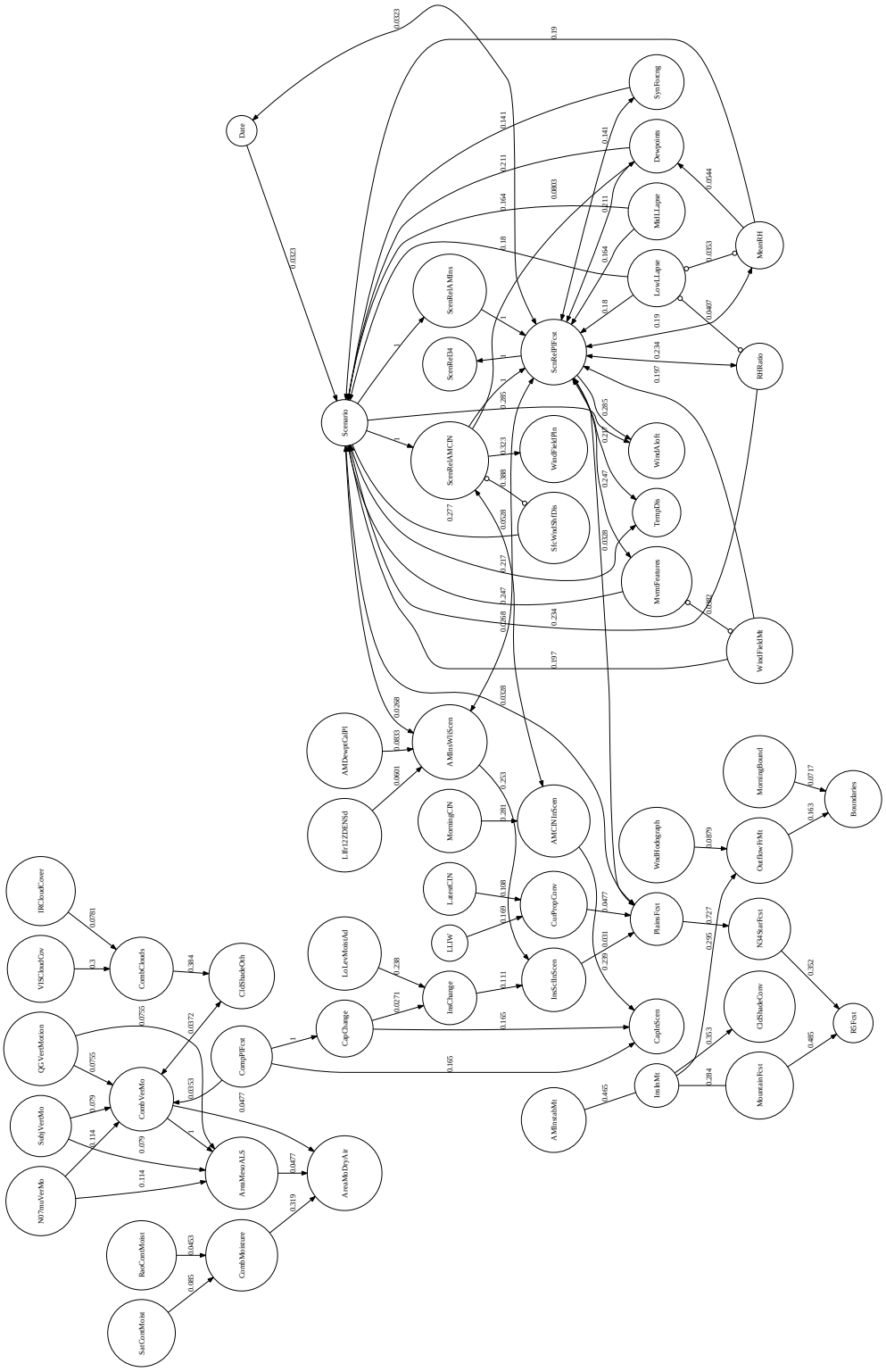


Figure B.29: Hailfinder Network Learnt by the LUMIN Learner with $NMI = 0.02$, $NCMI = 0.9$ with Heuristic Link Removal

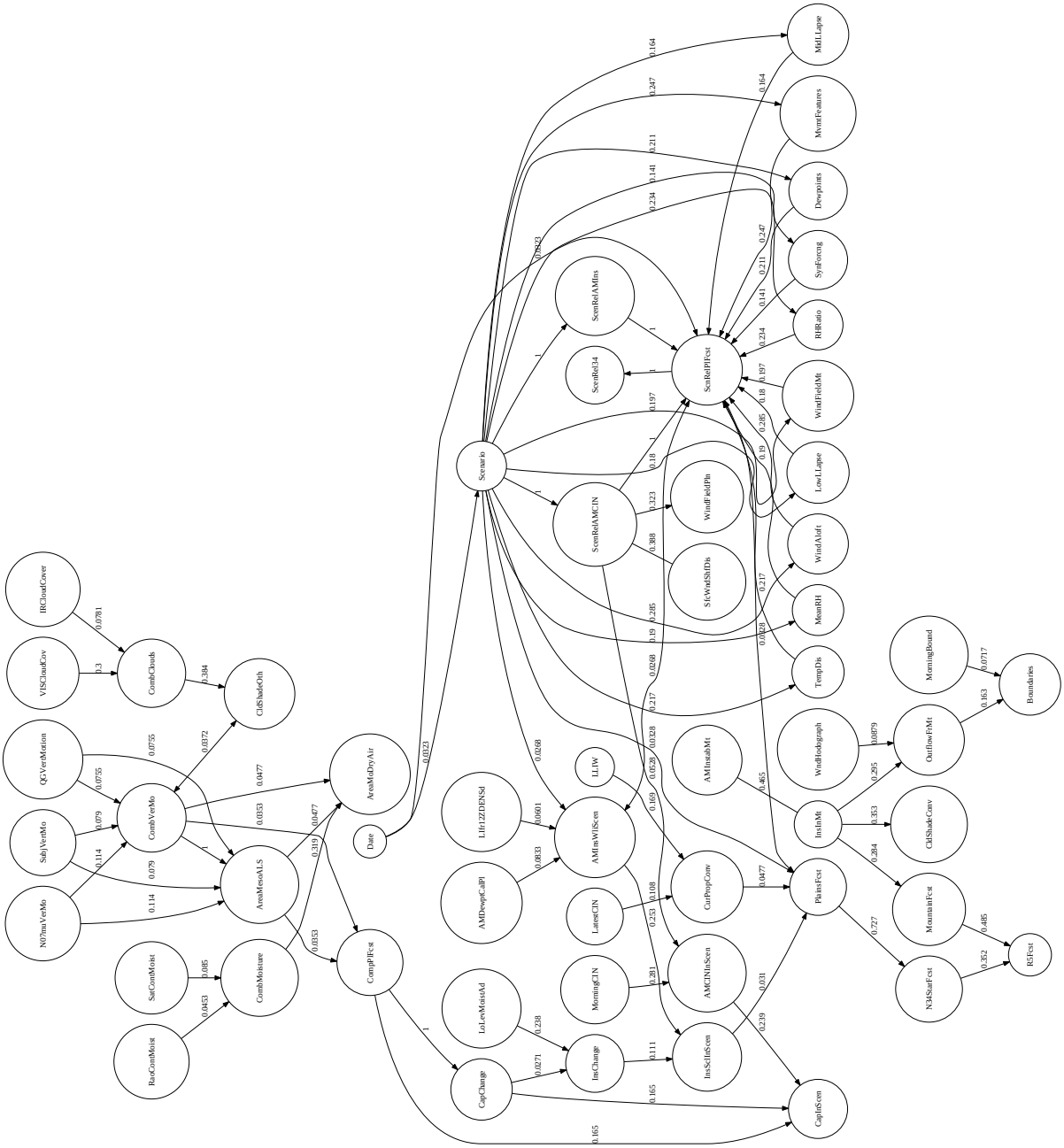


Figure B.30: Hailfinder network learnt by LUMIN with $NMI = 0.02$ and $NCMI = 0.7$ with Heuristic Link Removal and some Domain Information

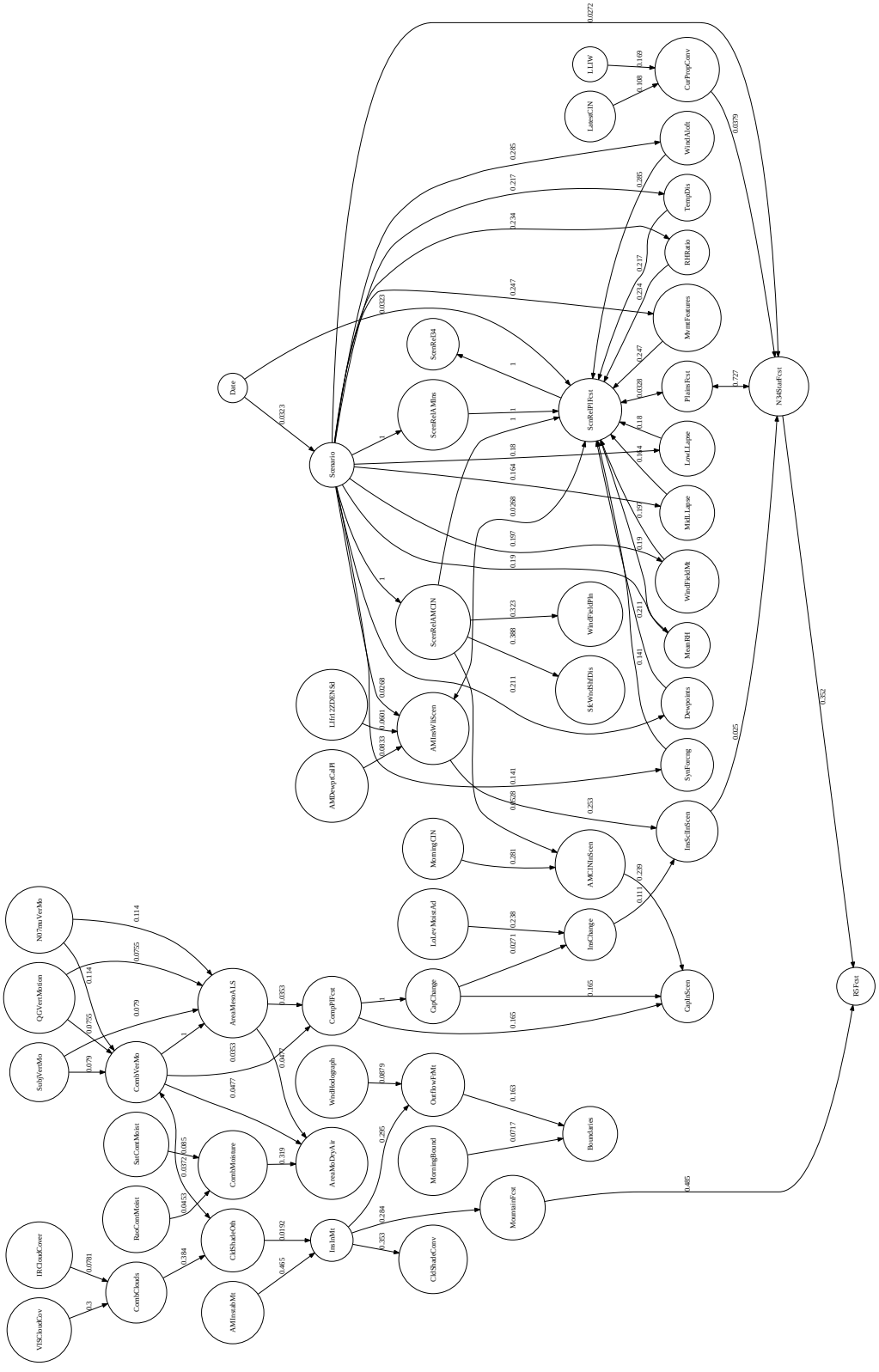


Figure B.31: Hailfinder Network Learnt by LUMIN with $NMI = 0.01$ and $NCMI = 0.9$ with Heuristic Link Removal and some Domain Information

The following graphs are from the Insurance network.

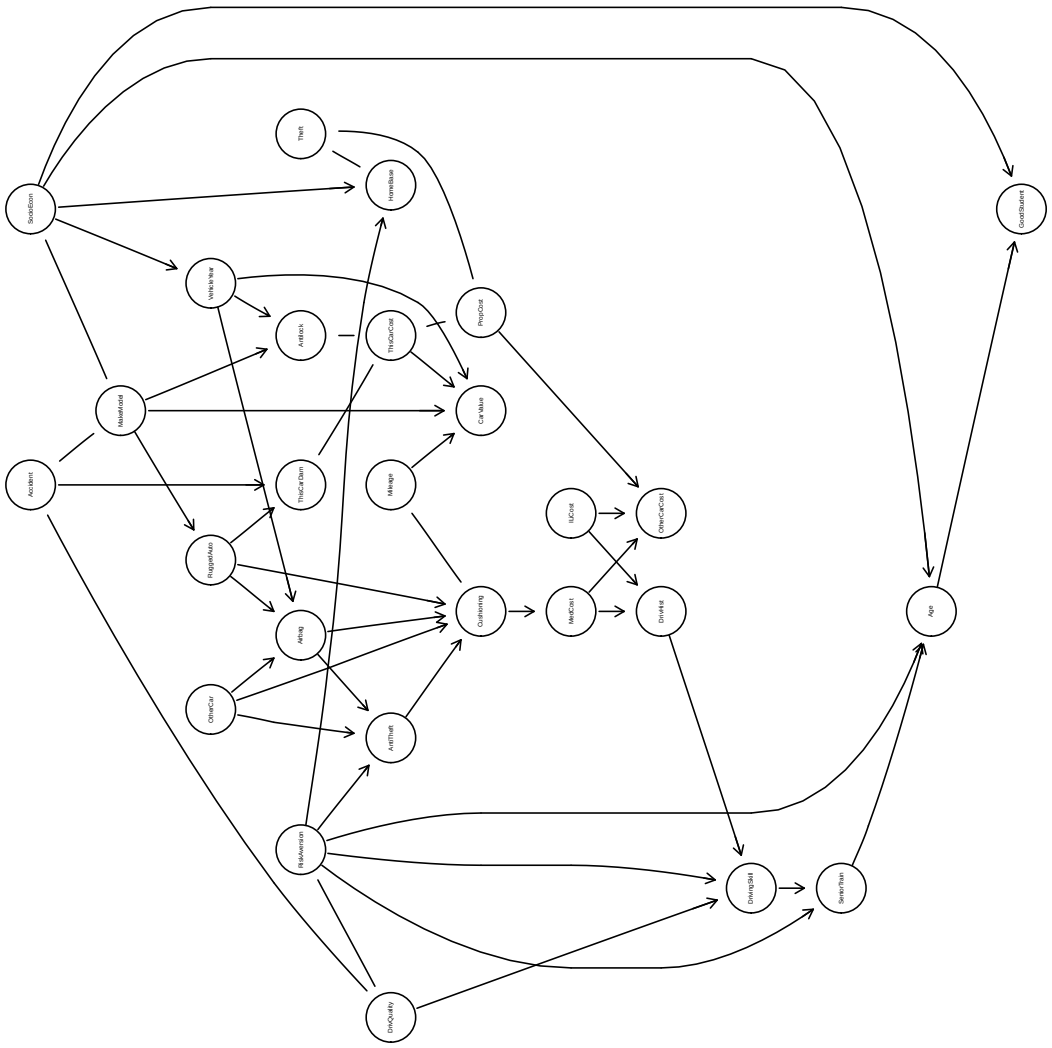


Figure B.32: Insurance Network Learnt by the Grow-Shrink Learner with $\alpha = 0.1$

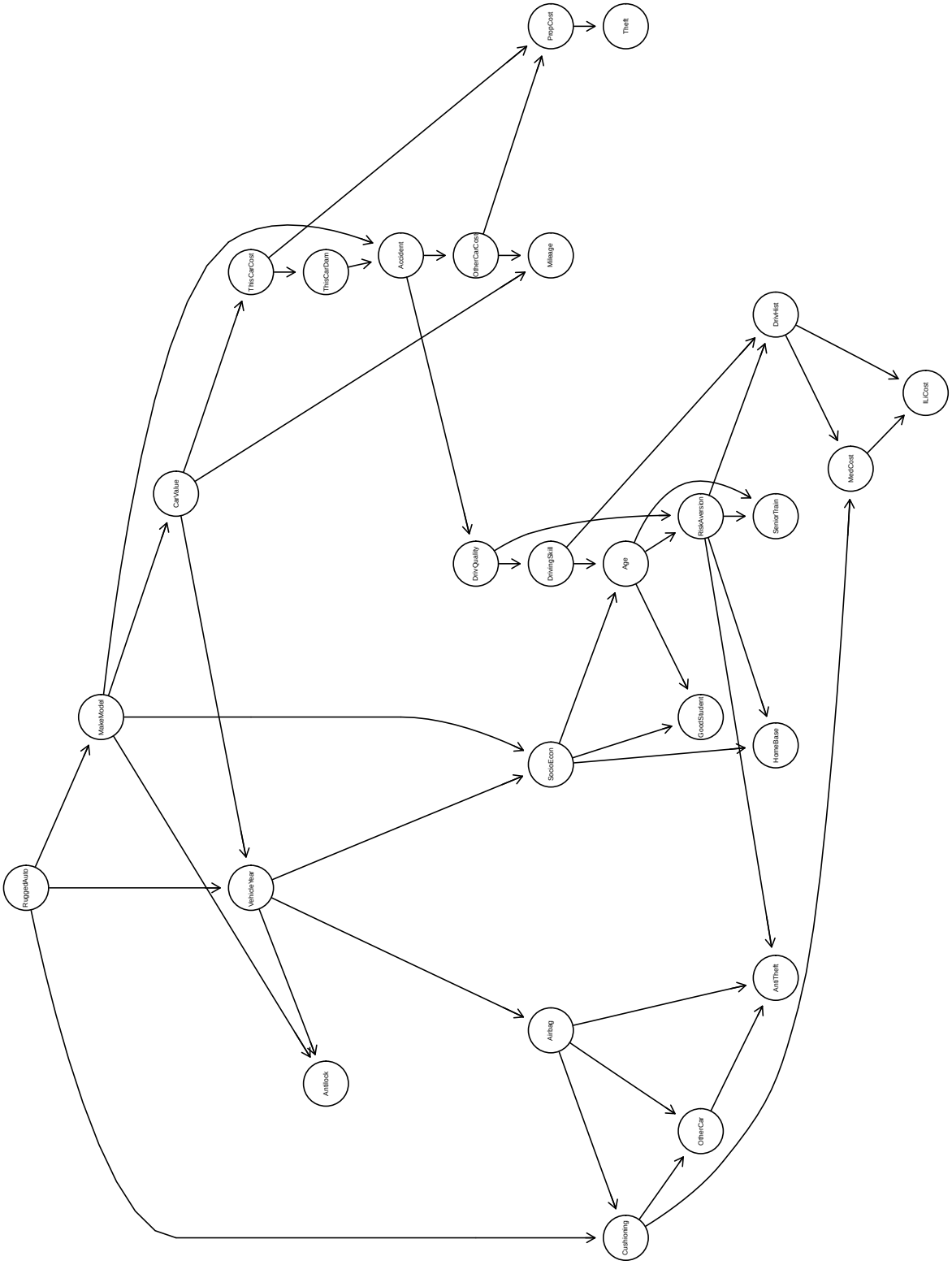


Figure B.33: Insurance Network Learnt by the Max-Min Hill Climbing Learner with $\alpha = 0.1$

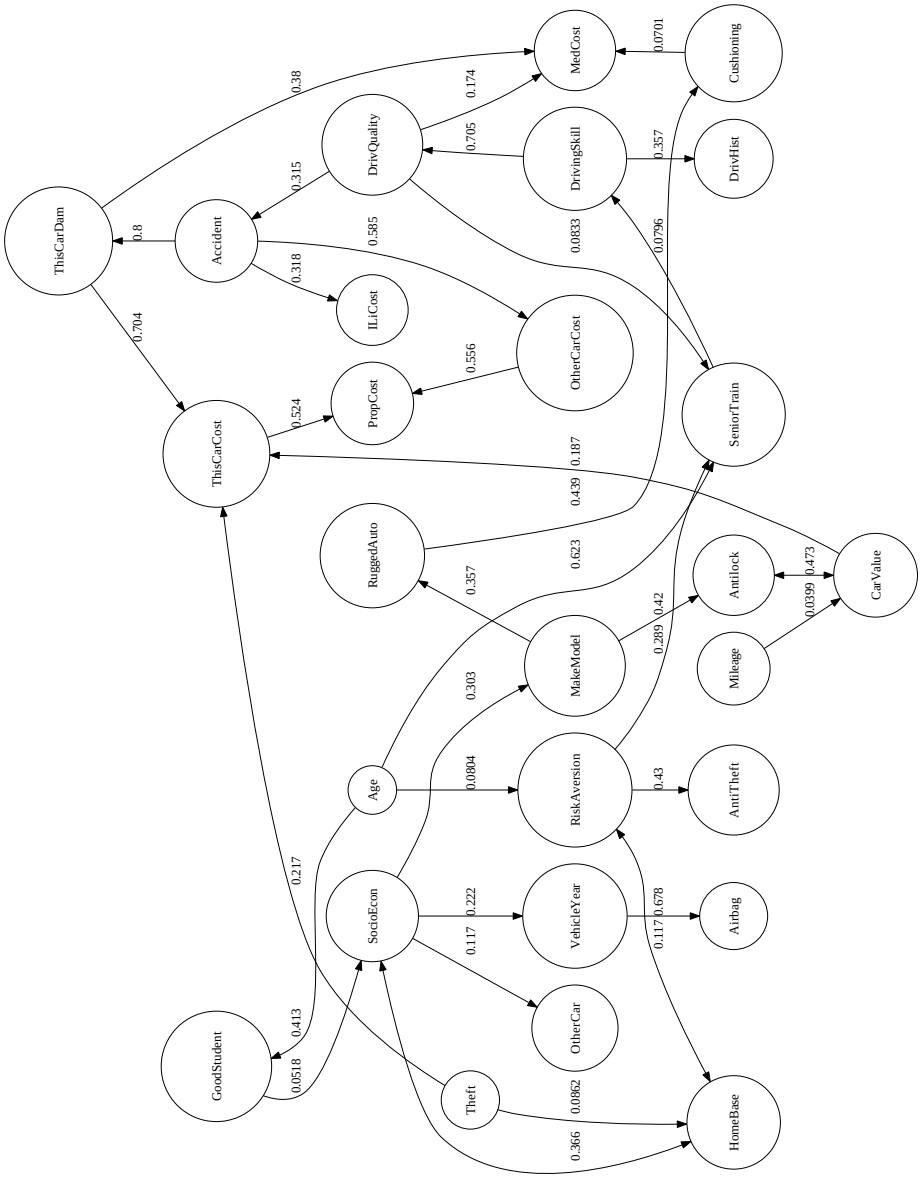


Figure B.34: Insurance Network Learnt by the LUMIN Learner with $NMI = 0.03$ and $NCMI = 0.7$ with Heuristic Link Removal and Limited Domain Information

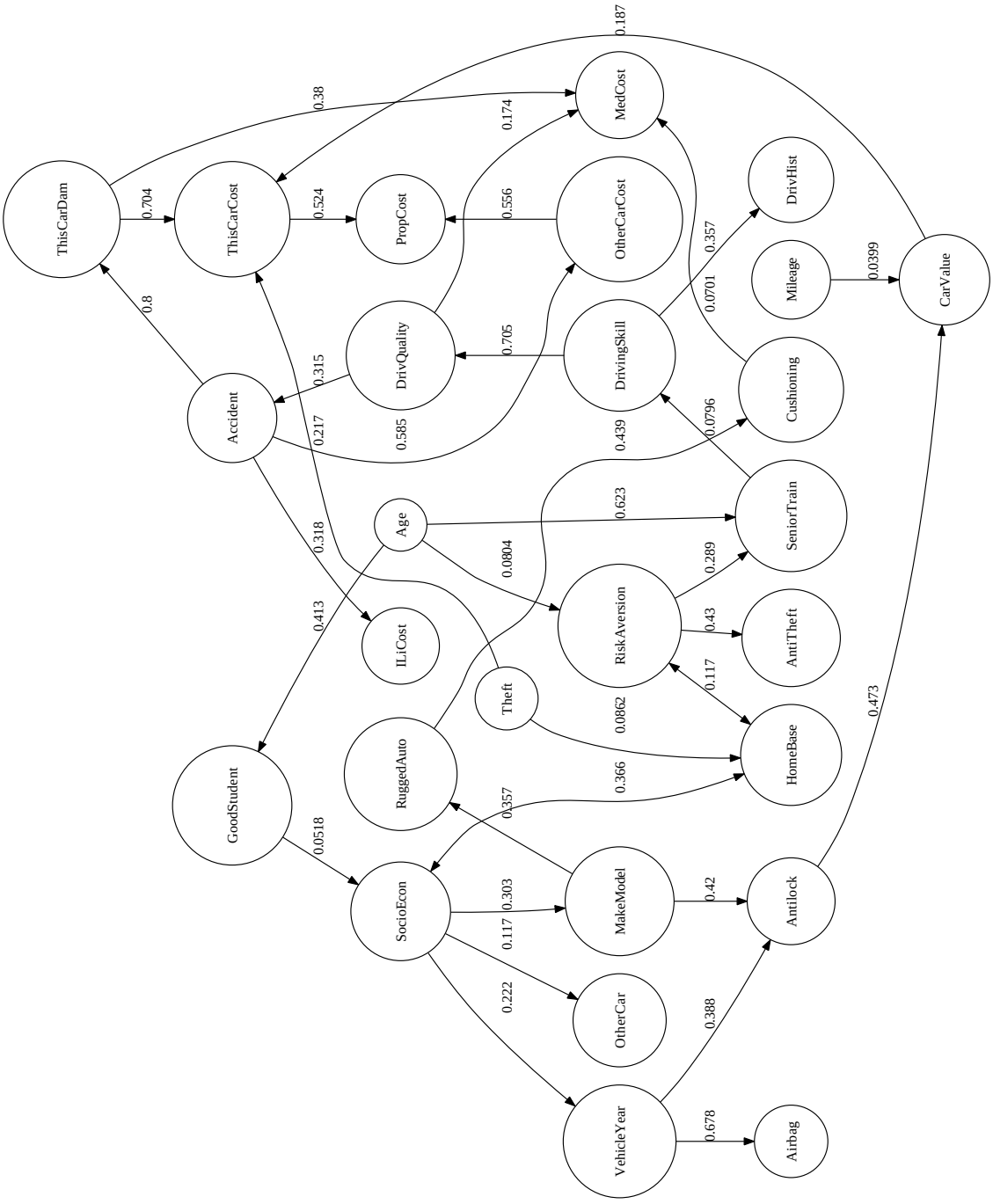


Figure B.35: Insurance Network Learnt by the LUMIN Learner with $NMI = 0.03$ and $NCMI = 0.9$ with Heuristic Link Removal and Limited Domain Information

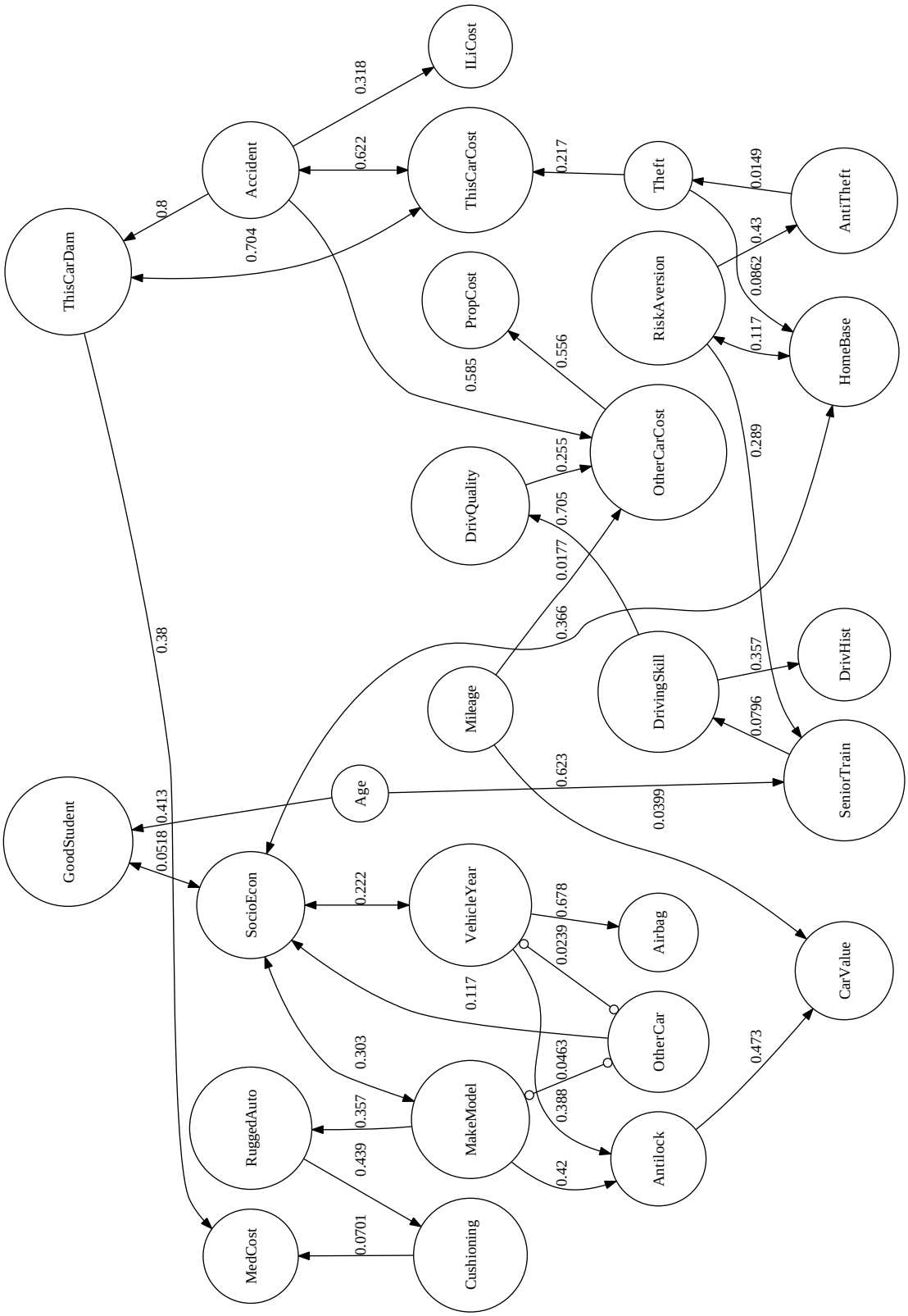


Figure B.36: Insurance Network Learnt by the LUMIN Learner with $NMI = 0.01$ and $NCMI = 0.9$ with Heuristic Link Removal and Limited Domain Information

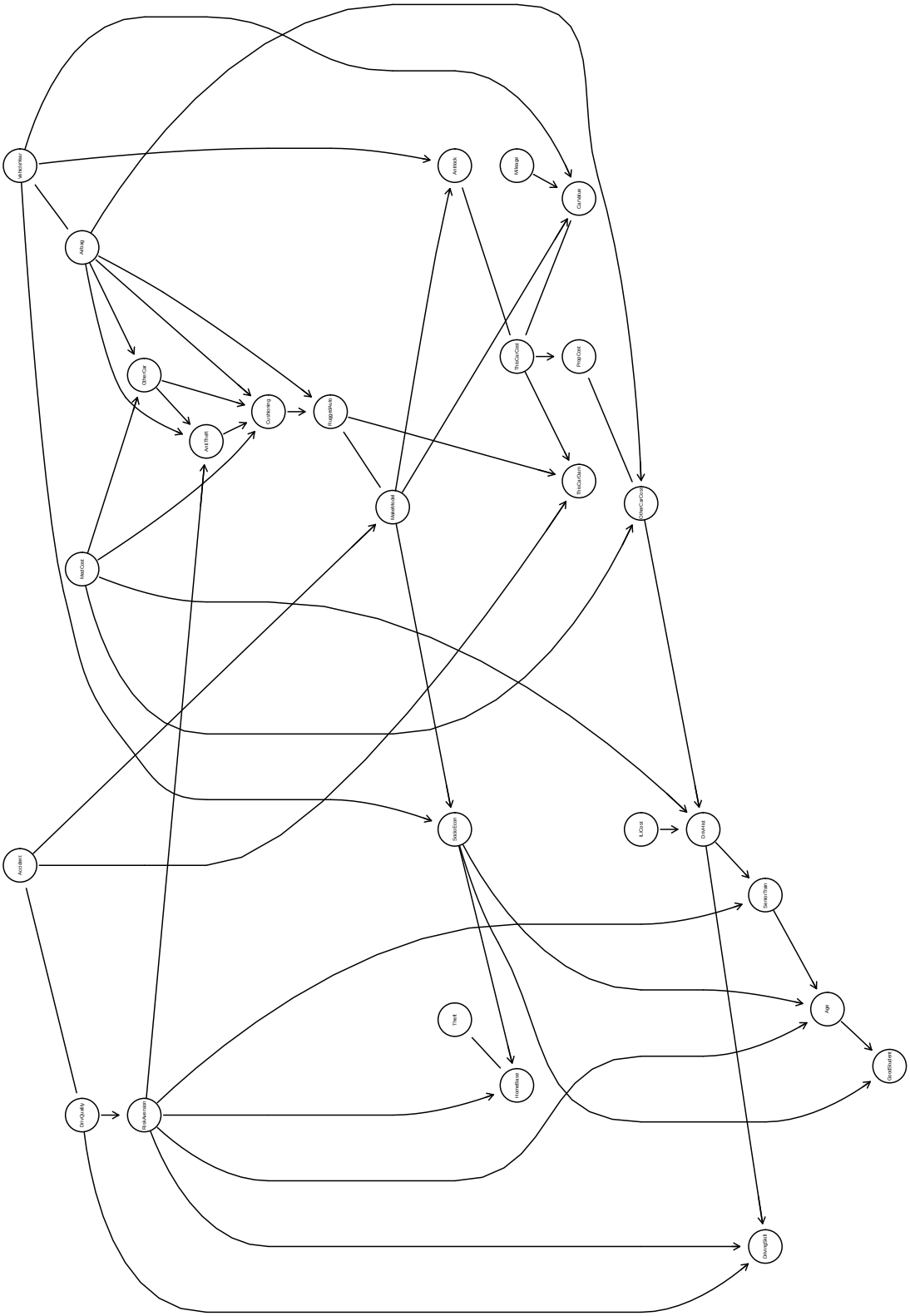


Figure B.37: Insurance Network Learnt by the Grow-Shrink Learner with $\alpha = 0.1$

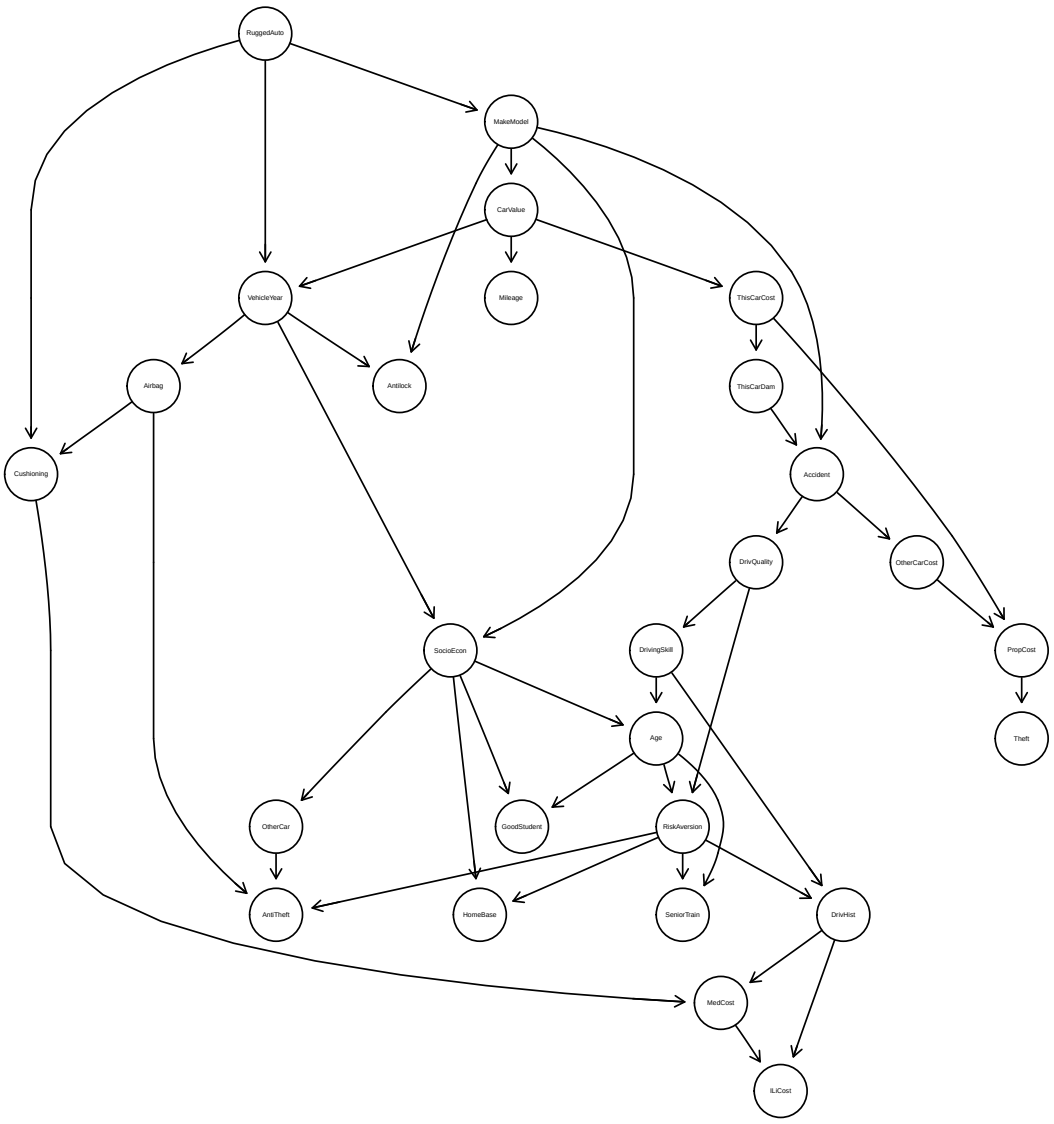


Figure B.38: Insurance Network Learnt by the Max-Min Hill Climbing Learner with $\alpha = 0.1$

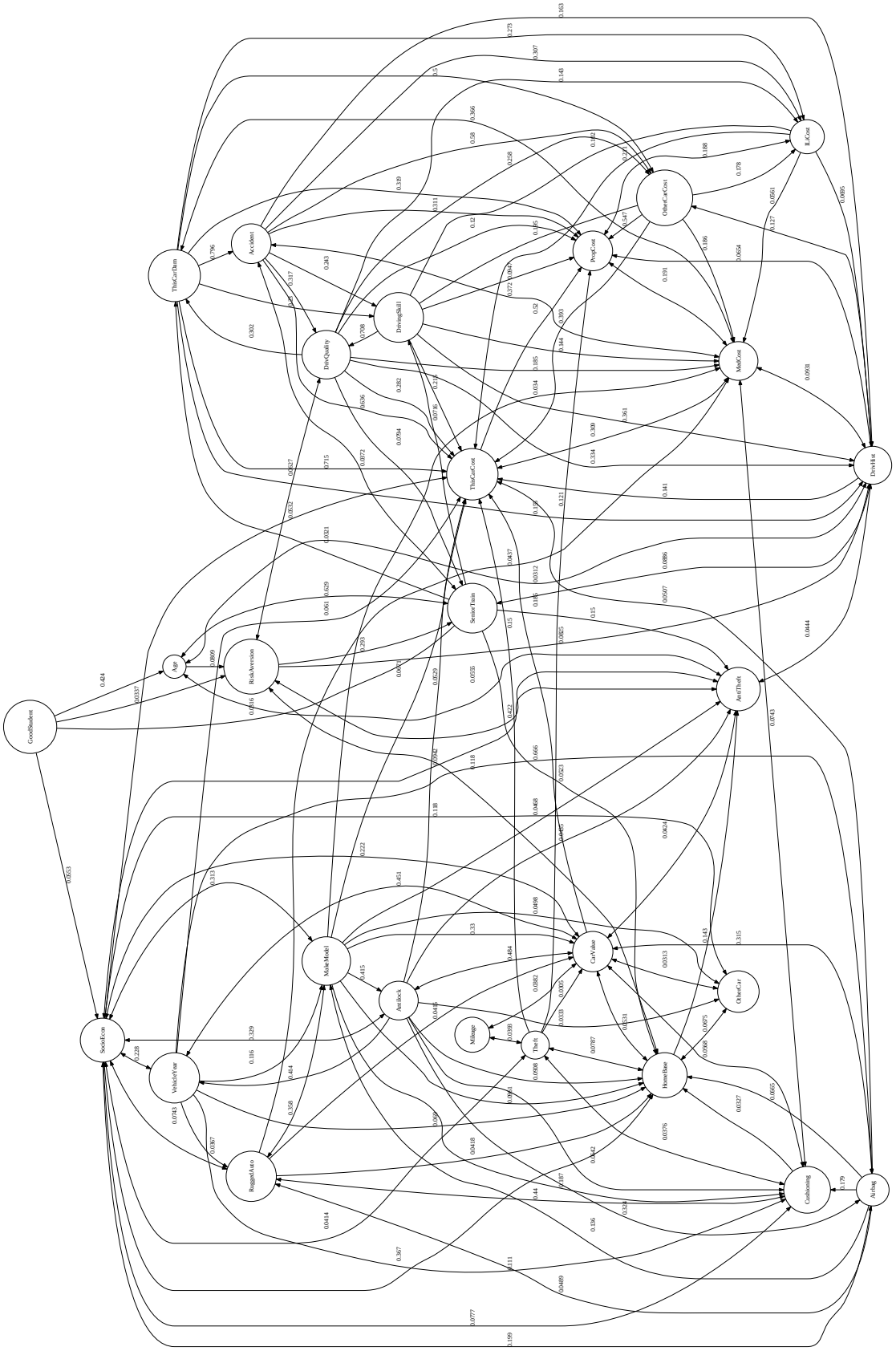


Figure B.39: Insurance Network Learnt by the LUMIN Learner with $NMI = 0.03$ and $NCMI = 0.5$ without Heuristic Link Removal

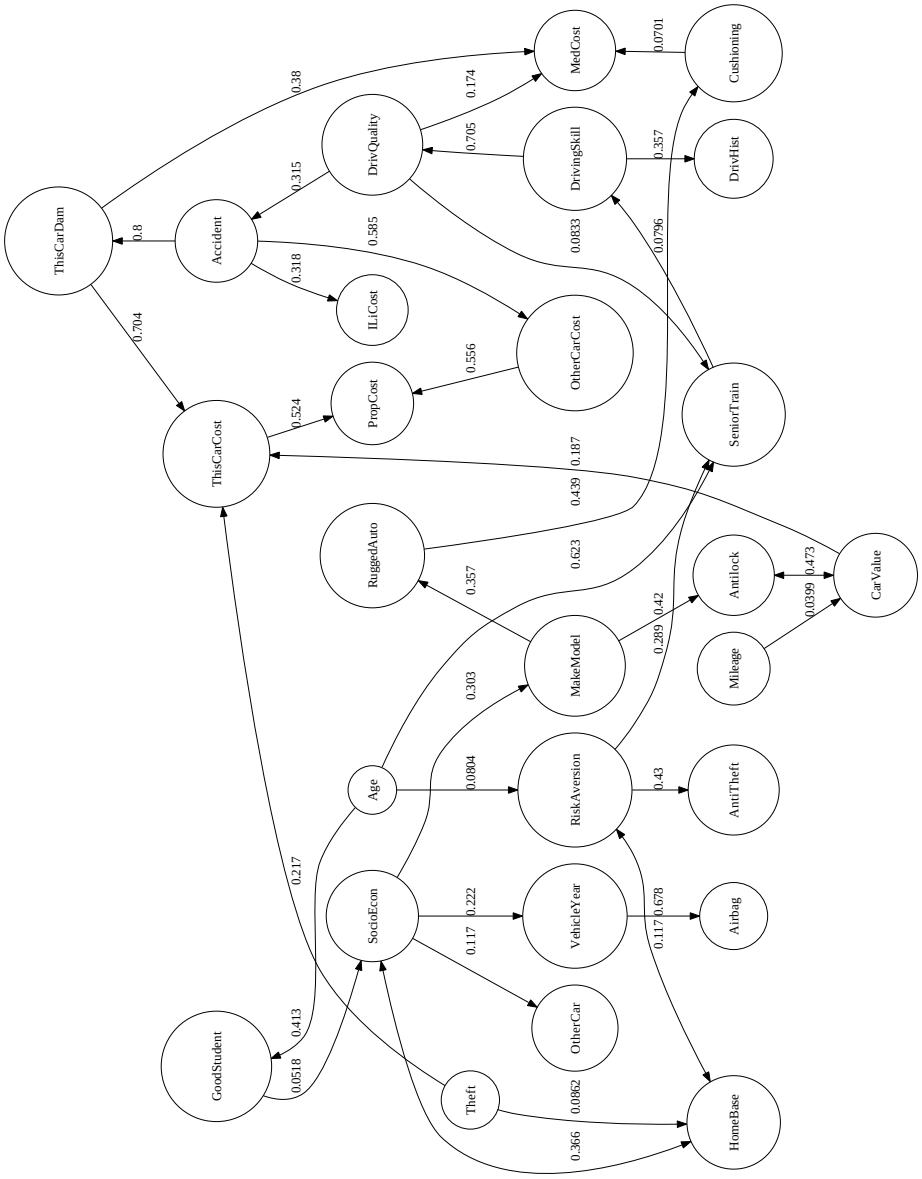


Figure B.40: Insurance Network Learnt by the LUMIN Learner with $NMI = 0.03$ and $NCMI = 0.7$ with Heuristic Link Removal and Domain Information

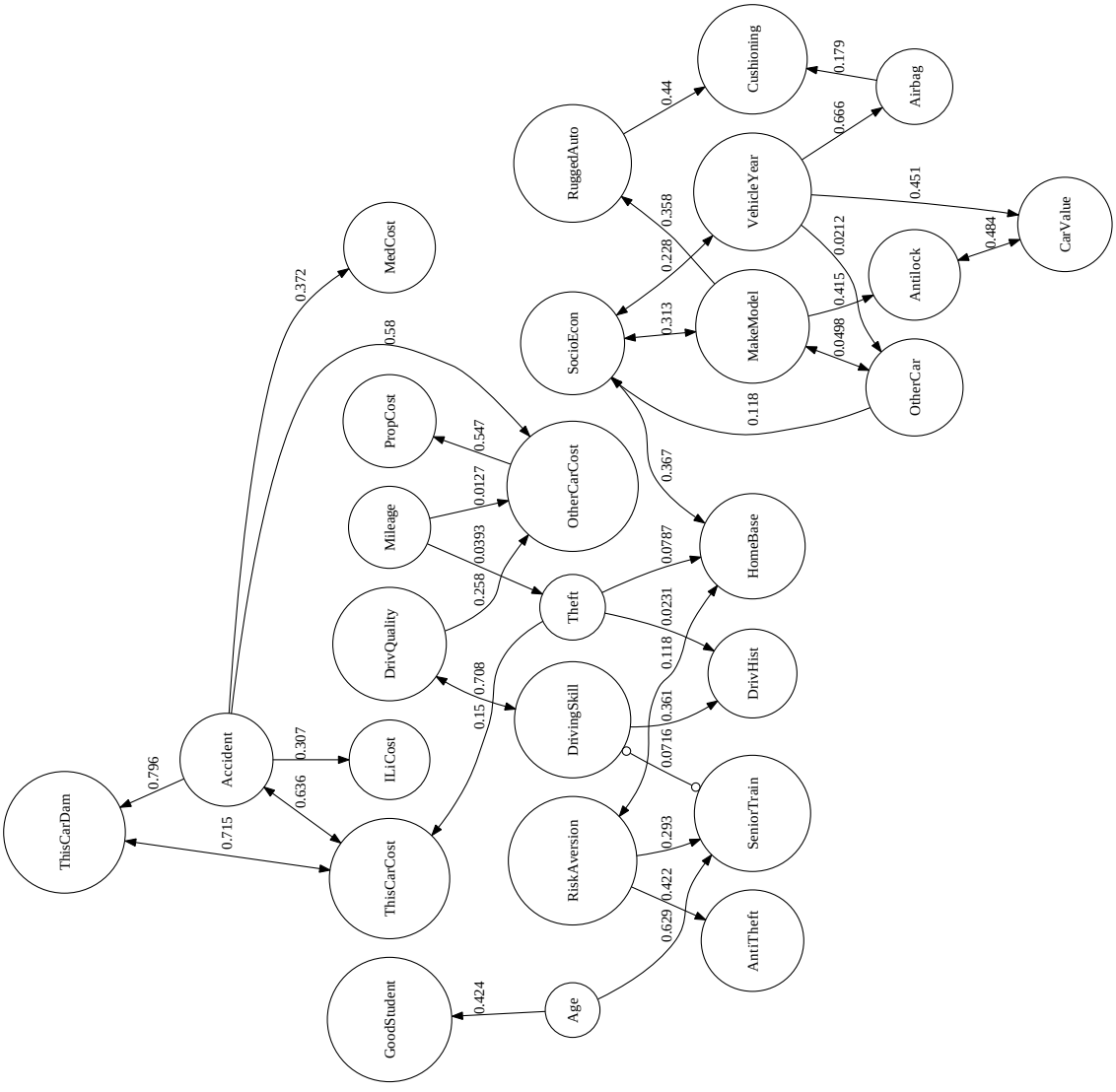


Figure B.41: Insurance Network Learnt by the LUMIN Learner with $NMI = 0.01$ and $NCMI = 0.9$ with Heuristic Link Removal and Domain Information

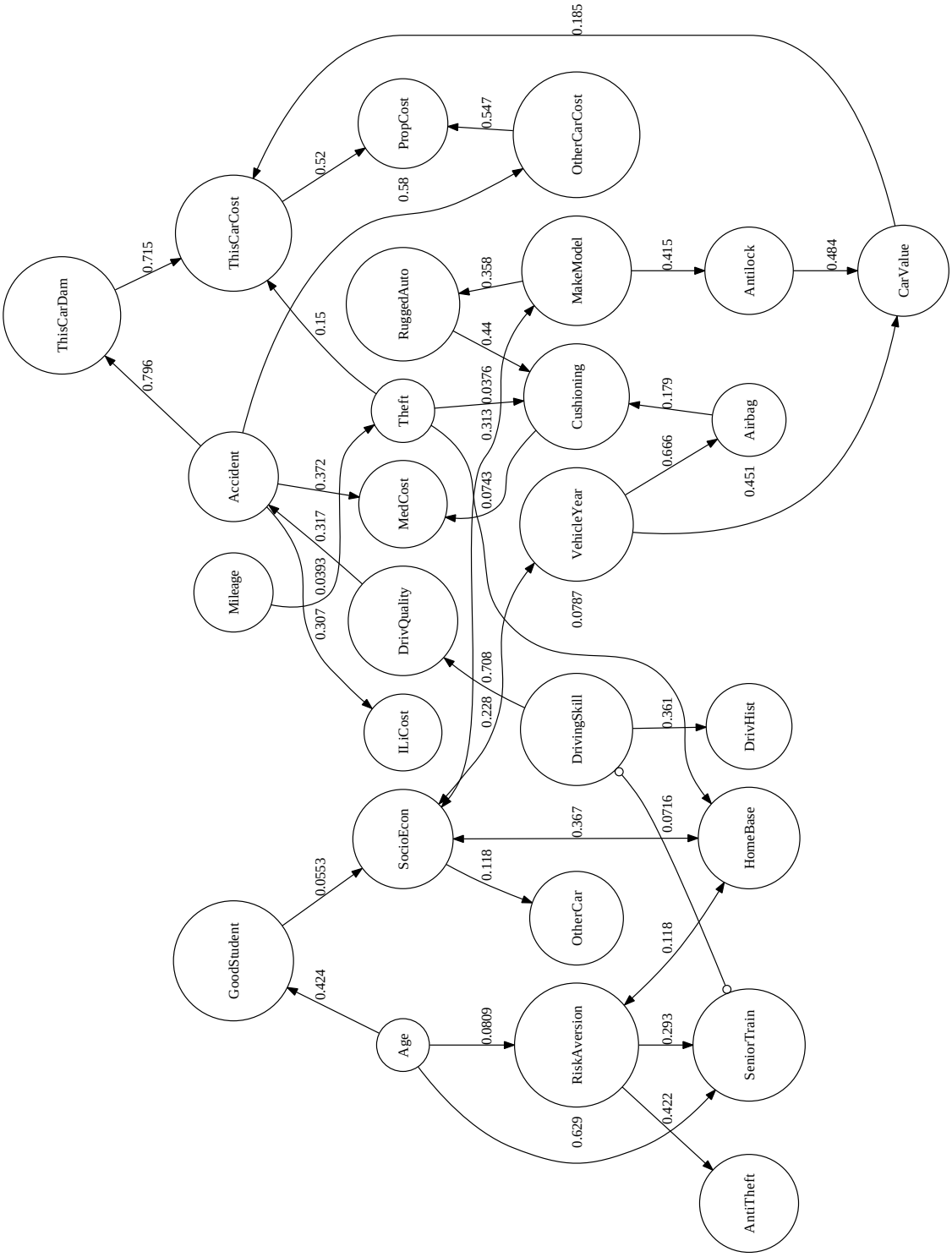


Figure B.42: Insurance Network Learnt by the LUMIN Learner with $NMI = 0.03$ and $NCMI = 0.9$ with Heuristic Link Removal and Domain Information

Appendix C

Triplets with Conflicts from the Hailfinder Dataset

Listed below in table C.1 are the triplets that gave rise to contradictory link directions for the LUMIN learner when analysing the Hailfinder dataset generated by the Bnlearn package.

Table C.1: Problem Triplets for LUMIN from the Bnlearn Hailfinder Dataset

CldShadeOth – CombVerMo – CompPlFcst	CldShadeOth – CombVerMo – CapChange
CldShadeOth – AreaMesoALS – CompPlFcst	CldShadeOth – AreaMesoALS – CapChange
CombVerMo – CldShadeOth – VISCloudCov	CombVerMo – CldShadeOth – IRCloudCover
CombVerMo – CldShadeOth – CombClouds	AreaMesoALS – CldShadeOth – VISCloudCov
AreaMesoALS – CldShadeOth – IRCloudCover	AreaMesoALS – CldShadeOth – CombClouds
AreaMoDryAir – CldShadeOth – VISCloudCov	AreaMoDryAir – CldShadeOth – IRCloudCover
AreaMoDryAir – CldShadeOth – CombClouds	AMInsWliScen – Scenario – RHRatio
PlainsFcst – Scenario – MeanRH	N34StarFcst – Scenario – RHRatio
LowLLapse – Scenario – SynForcng	MeanRH – Scenario – SynForcng
TempDis – Scenario – WindFieldMt	AMCINInScen – ScenRelAMCIN – PlainsFcst
AMCINInScen – ScenRelAMCIN – N34StarFcst	ScenRel34 – ScenRelAMCIN – WindAloft
PlainsFcst – ScenRelAMCIN – Dewpoints	PlainsFcst – ScenRelAMCIN – LowLLapse
PlainsFcst – ScenRelAMCIN – MeanRH	PlainsFcst – ScenRelAMCIN – RHRatio
N34StarFcst – ScenRelAMCIN – Dewpoints	N34StarFcst – ScenRelAMCIN – LowLLapse
N34StarFcst – ScenRelAMCIN – MeanRH	N34StarFcst – ScenRelAMCIN – RHRatio

LowLLapse – ScenRelAMCIN – SynForcng	LowLLapse – ScenRelAMCIN – WindAloft
MeanRH – ScenRelAMCIN – WindAloft	RHRatio – ScenRelAMCIN – WindAloft
SynForcng – ScenRelAMCIN – TempDis	TempDis – ScenRelAMCIN – WindAloft
Scenario – AMCINInScen – MorningCIN	ScenRelAMCIN – AMCINInScen – MorningCIN
Date – ScenRelAMIns – AMInsWliScen	Date – ScenRelAMIns – PlainsFcst
Date – ScenRelAMIns – N34StarFcst	ScenRel34 – ScenRelAMIns – MvmtFeatures
ScenRel34 – ScenRelAMIns – SynForcng	ScenRel34 – ScenRelAMIns – TempDis
ScenRel34 – ScenRelAMIns – WindAloft	ScenRel34 – ScenRelAMIns – WindFieldMt
PlainsFcst – ScenRelAMIns – MeanRH	PlainsFcst – ScenRelAMIns – WindFieldMt
LowLLapse – ScenRelAMIns – SynForcng	LowLLapse – ScenRelAMIns – WindAloft
MeanRH – ScenRelAMIns – SynForcng	MidLLapse – ScenRelAMIns – MvmtFeatures
MidLLapse – ScenRelAMIns – RHRatio	MidLLapse – ScenRelAMIns – SynForcng
MidLLapse – ScenRelAMIns – TempDis	MidLLapse – ScenRelAMIns – WindAloft
MvmtFeatures – ScenRelAMIns – WindAloft	RHRatio – ScenRelAMIns – WindAloft
SfcWndShfDis – ScenRelAMIns – WindFieldMt	WindFieldMt – ScenRelAMIns – SynForcng
TempDis – ScenRelAMIns – WindAloft	TempDis – ScenRelAMIns – WindFieldMt
WindFieldMt – ScenRelAMIns – WindAloft	Scenario – AMInsWliScen – LIfr12ZDENSd
Scenario – AMInsWliScen – AMDewptCalPl	ScenRelAMIns – AMInsWliScen – LIfr12ZDENSd
ScenRelAMIns – AMInsWliScen – AMDewptCalPl	ScenRelAMCIN – ScenRel34 – MidLLapse
ScenRelAMCIN – ScenRel34 – MvmtFeatures	ScenRelAMCIN – ScenRel34 – RHRatio
ScenRelAMCIN – ScenRel34 – SynForcng	ScenRelAMCIN – ScenRel34 – TempDis
ScenRelAMCIN – ScenRel34 – WindAloft	ScenRelAMCIN – ScenRel34 – WindFieldMt
ScenRelAMIns – ScenRel34 – MvmtFeatures	ScenRelAMIns – ScenRel34 – SynForcng
ScenRelAMIns – ScenRel34 – TempDis	ScenRelAMIns – ScenRel34 – WindAloft
ScenRelAMIns – ScenRel34 – WindFieldMt	Date – ScnRelPlFcst – AMCINInScen
Date – ScnRelPlFcst – AMInsWliScen	Date – ScnRelPlFcst – PlainsFcst
Date – ScnRelPlFcst – N34StarFcst	AMCINInScen – ScnRelPlFcst – AMInsWliScen
AMCINInScen – ScnRelPlFcst – PlainsFcst	AMCINInScen – ScnRelPlFcst – N34StarFcst
AMCINInScen – ScnRelPlFcst – MidLLapse	AMCINInScen – ScnRelPlFcst – MvmtFeatures
AMCINInScen – ScnRelPlFcst – WindFieldMt	AMInsWliScen – ScnRelPlFcst – PlainsFcst

AMInsWliScen – ScnRelPlFcst – N34StarFcst	AMInsWliScen – ScnRelPlFcst – MidLLapse
AMInsWliScen – ScnRelPlFcst – MvmtFeatures	AMInsWliScen – ScnRelPlFcst – RHRatio
AMInsWliScen – ScnRelPlFcst – SynForcng	AMInsWliScen – ScnRelPlFcst – WindFieldMt
PlainsFcst – ScnRelPlFcst – Dewpoints	PlainsFcst – ScnRelPlFcst – MeanRH
PlainsFcst – ScnRelPlFcst – MvmtFeatures	N34StarFcst – ScnRelPlFcst – Dewpoints
N34StarFcst – ScnRelPlFcst – MvmtFeatures	N34StarFcst – ScnRelPlFcst – RHRatio
LowLLapse – ScnRelPlFcst – SynForcng	MeanRH – ScnRelPlFcst – SynForcng
MidLLapse – ScnRelPlFcst – SynForcng	TempDis – ScnRelPlFcst – WindFieldMt
Scenario – PlainsFcst – InsSclInScen	Scenario – PlainsFcst – CurPropConv
Scenario – PlainsFcst – N34StarFcst	ScenRelAMCIN – PlainsFcst – InsSclInScen
ScenRelAMCIN – PlainsFcst – CurPropConv	ScenRelAMIns – PlainsFcst – InsSclInScen
ScenRelAMIns – PlainsFcst – CurPropConv	InsSclInScen – PlainsFcst – ScnRelPlFcst
CurPropConv – PlainsFcst – ScnRelPlFcst	ScnRelPlFcst – PlainsFcst – N34StarFcst
Scenario – N34StarFcst – InsSclInScen	Scenario – N34StarFcst – CurPropConv
Scenario – N34StarFcst – PlainsFcst	ScenRelAMCIN – N34StarFcst – InsSclInScen
ScenRelAMCIN – N34StarFcst – CurPropConv	ScenRelAMIns – N34StarFcst – InsSclInScen
ScenRelAMIns – N34StarFcst – PlainsFcst	InsSclInScen – N34StarFcst – ScnRelPlFcst
CurPropConv – N34StarFcst – ScnRelPlFcst	ScnRelPlFcst – N34StarFcst – PlainsFcst
OutflowFrMt – R5Fcst – PlainsFcst	OutflowFrMt – R5Fcst – N34StarFcst
CldShadeConv – R5Fcst – PlainsFcst	CldShadeConv – R5Fcst – N34StarFcst
ScenRelAMCIN – SynForcng – MvmtFeatures	ScenRelAMCIN – SynForcng – WindAloft
ScenRelAMCIN – SynForcng – WindFieldMt	SfcWndShfDis – SynForcng – WindFieldMt
ScenRelAMCIN – WindAloft – MvmtFeatures	ScenRelAMCIN – WindAloft – WindFieldMt
ScenRelAMIns – WindAloft – WindFieldMt	MvmtFeatures – WindAloft – SfcWndShfDis
SfcWndShfDis – WindAloft – WindFieldMt	

Appendix D

Source Code and Data

The source of the LUMIN program is available at `http://www.eecs.qmul.ac.uk/~norman/Joseph/LUMIN.tgz`.

The datasets used in the thesis are available from `http://www.eecs.qmul.ac.uk/~norman/Joseph/Data.tgz`.

Updated versions of the LUMIN program and more details on the generation of the non-standard datasets can be obtained from the author contact `adrian@dragons-joseph.org`.

Index

Bibliography

[Abramson et al 1996] Abramson B., Brown J., Edwards W., Murphy A., and Winkler R.L. (1996). Hailfinder: A Bayesian system for forecasting severe weather. International Journal of Forecasting, volume 12, issue 1, pp. 57-71. Elsevier B. V.

[Acorn & Walden 1992] Acorn T. L. and Walden S. H. (1992). SMART: Support management automated reasoning technology for Compaq customer service. Proceedings of the Fourth Conference on Innovative Applications of Artificial Intelligence, pp. 3-18. AAAI Press.

[Aha 1992] Aha D. W. (1992). Tolerating noisy, irrelevant and novel attributes in instance based learning algorithms. International Journal of Man-Machine Studies, 36, pp 267-287.

[Aizerman, Braverman & Rozonoér 1964] Aizerman M. A., Braverman É. M., and Rozonoér. (1964). Theoretical foundations of the potential function method in pattern recognition learning. Automation and Remote Control 25. pp 821-837.

[Aliferis and Cooper 1994] Aliferis C., and Cooper G. (1994). An evaluation of an algorithm for inductive learning of Bayesian belief networks using simulated data sets. Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence, pp. 8-14. Morgan Kaufmann.

[Allen 1993] Allen B. (1993). Truth in Philosophy. Harvard University Press, Cambridge, Massachsets, USA.

[Almuallim & Dietterich 1991] Almuallim H., & Dietterich T. G. (1991). Learning with many irrelevant features. In Proceedings of the Ninth National Conference on Artificial Intelligence, pp. 547-552. Menlo Park, CA: AAAI Press.

[Anderson & Belnap 1975] Anderson A.R. and Belnap N.D. (1975). Entailment: The Logic of Relevance and Necessity, volume I. Princeton University Press.

[Anderson, Belnap & Dunn 1992] Anderson A.R., Belnap N.D. and Dunn J.M. (1992). Entailment: The Logic of Relevance and Necessity, volume II. Princeton University Press.

[Andrews et al 1995] Andrews R., Diederich J. & Tickle A. (1995). A Survey and Critique of Techniques For Extracting Rules From Trained Artificial Neural Networks. Knowledge Based Systems, volume 8, pp. 373-389. Elsevier.

[Andrieu et al 2001] Andrieu C., de Freitas N., & Doucet A. (2001). Robust full Bayesian learning for radial basis networks. Neural Computation, volume 13, issue 10, 2359-2407. MIT Press Cambridge, MA, USA.

[Anthony & Bartlett 1999] Anthony M., and Bartlett P. (1999). Neural Network Learning: Theoretical Foundations. Cambridge University Press.

[Aristotle] Aristotle (1933-35). Metaphysics. Translated by Hugh Tredennick in two volumes. Loeb Classical Library 271, 287. Harvard University Press.

[Arnauld & Nicole 1662] Arnauld A., and Nicole P. (1662). Logic, or, The Art of Thinking ("The Port Royal Logic"). Translated by J. Dickoff and P. James. Indianapolis: Bobbs-Merrill, 1964.

[Bacon 1620] Bacon F. (1620). The New Organon, Lisa Jardine & Michael Silverthorne (eds), Cambridge University Press 2000.

[Baker 1975] Baker J. K. (1975). "The DRAGON System–An Overview. IEEE Transactions on Acoustics, Speech, and Signal Processing, volume 23, issue 1, pp. 24-29. IEEE Press.

[Baldwin 1896] Baldwin J.M. (1896). A New Factor in Evolution. The American Naturalist, volume 30, number 354, pp. 441-451. The University of Chicago Press.

[Bareiss 1989] Bareiss E. R. (1989). Exemplar-based knowledge acquisition: A unified approach to concept representation classification and learning. Boston, Academic Press.

[Barletta & Hennessy 1989] Barletta R. and Hennessy D. (1989). Case adaptation in autoclave layout design. Proceedings: Case-Based Reasoning Workshop. Morgan Kaufmann Publishers.

[Bartle 1995] Bartle R.G. (1995). The Elements of Integration and Lebesgue Measure. Wiley Interscience.

[Bay et al 2002] Bay S.D., Shapiro D.G., and Langley P. (2002). Revising Engineering Models: Combining Computational Discovery with Knowledge. Proceedings of the Thirteenth European Conference on Machine Learning, ECML 2002, pp. 10-22. Springer-Verlag London, UK.

[Beinlich et al 1989] Beinlich I.A., Suermondt H. J., Chavez R. M., and Cooper G. F. (1989). The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks. Proceedings of the Second European Conference on Artificial Intelligence in Medicine (London, Aug.), 38, pp 247-256.

[Bell and Wang 2000] Bell D.A. and Wang H. (2000) A formalism for relevance and its application in feature subset selection. Machine Learning, 41, pp 175-195.

[Ben-Bassat 1982] Ben-Bassat M. (1982). Pattern recognition and reduction of dimensionality. In P.R. Krishnaiah, L.N. Kanal, eds, Handbook of statistics 2: classification, pattern recognition and reduction of dimensionality. North Holland, Amsterdam.

[Bengio et al 2007] Bengio Y., Lamblin P., Popovici P., and Larochelle H. (2007). Greedy Layer-Wise Training of Deep Networks, Advances in Neural Information Processing Systems, volume 19, pp. 153-160. MIT Press, Cambridge, MA.

[Berger 1985] Berger J. (1985). Statistical Decision Theory and Bayesian Analysis. Springer Verlag, New York.

[Bernardo and Smith 1994] Bernardo J.M., Smith A.F.M. (1994). Bayesian Theory. Chichester: Wiley.

[Bertrand 1889] Bertrand J. L. F. (1889). Calcul des probabilités. Paris: Gauthier-Villars et fils.

[Binder et al 1997] Binder J., Koller D., Russell S., and Kanazawa K. (1997). Adaptive Probabilistic Networks with Hidden Variables. Machine Learning, volume 29, number 2-3, pp. 213-244. Springer Netherlands.

[Bishop 1996] Bishop C. M. (1996). Neural networks for pattern recognition. Oxford, England: Oxford University Press.

[Bishop & Svensén 2003] Bishop C. M. & Svensén M. (2003). Bayesian Hierarchical Mixtures of Experts. Proceedings Nineteenth Conference on Uncertainty in Artificial Intelligence, pp. 57-64. Morgan Kaufmann.

[Blockeel and De Raedt 1998] Blockeel H. and De Raedt L. (1998). Top-down Induction of Logical Decision Trees. Artificial Intelligence, volume 101, pp 285-297. Elsevier Science Publishers B. V. Amsterdam, The Netherlands.

[Blockeel et al 2005] Blockeel H., Page D., and Srinivasan A. (2005). Multi-Instance Tree Learning. In Proceedings of the twenth second International Conference on Machine Learning, pp 57-64. ACM Press.

[Bobrow 1964] Bobrow D. G. (1964). Natural Language Input for a Computer Problem Solving System. PhD Thesis at Massachusetts Institute of Technology. `http://hdl.handle.net/1721.1/5922`

[Boerlage 1994] Boerlage B. (1994). Link Strength in Bayesian Networks. Technical Report: TR-94-17, University of British Columbia Vancouver, BC, Canada.

[Bongard & Lipson 2005] Bongard J. and Lipson H. (2005). Active Coevolutionary Learning of Deterministic Finite Automata. Journal of Machine Learning Research, volume 6, pp. 1651-1678. MIT Press Cambridge, MA, USA.

[Boser, Guyon & Vapnik 1992] Boser B. M., Guyon I. M., Vapnik V. N. (1992). A Training Algorithm for Optimal Margin Classifiers. Proceedings of 5th Anual ACM Workshop on Computational Learning Theory, pp 144-152. ACM Press.

[Bradley 1914] Bradley F.H. (1914). Essays on Truth and Reality. Clarendon press, Oxford, England.

[Bregman 1967] Bregman L. (1967). The Relaxation Method of Finding the Common Point of Convex Sets and Its Application to the Solution of Problems in Convex Programming. USSR

Computational Mathematics and Mathematical Physics, volume 7, pp. 200-217. Pleiades Publishing.

[Breiman 1996] Breiman L. (1996). Bagging Predictors. Machine Learning, volume 24, (2), pp 123-140. Springer Netherlands.

[Broomhead and Lowe 1988] Broomhead D.S. and Lowe D. (1988). Multivariate functional interpolation and adaptive networks. Complex Systems, 2, pp 321-355.

[Buchanan & Shortliffe 1984] Buchanan B. G. and Shortliffe E. H. (1984). Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project. Reading, MA: Addison-Wesley.

[Buntine 1991] Buntine W. (1991). Theory refinement on Bayesian networks. Proceedings of the seventh conference on Uncertainty in artificial intelligence (UAI 91), pp. 52-60. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA.

[Burck 1965] Burck G. (1965). The computer age and its potential for management. Harper & Row.

[Cameron-Jones & Quinlan 1993] Cameron-Jones R., & Quinlan J. R. (1993). Avoiding pitfalls when learning recursive theories. Proceedings of the Eighth International Workshop on Machine Learning (pp. 389-393). San Mateo, CA: Morgan Kaufmann.

[de Campos 2006] de Campos L.M. (2006). A Scoring Function for Learning Bayesian Networks based on Mutual Information and Conditional Independence Tests. The Journal of Machine Learning Research, volume 7, pp. 2149-2187. JMLR.org.

[Carbonell et al 1990] Carbonell J, Knoblock C, & Minton S. (1990). PRODIGY: An integrated architecture for planning and learning. In K. VanLehn (Ed.), Architectures for Intelligence. Hillsdale, NJ: Lawrence Erlbaum Associates.

[Carnap 1950] Carnap R. (1950). Logical Foundations of Probability. University of Chicago Press.

[Carnap 1963] Carnap R. (1963). Replies and Systematic Expositions. The Philosophy of Rudolf Carnap. Open Court, La Salle, Illinois, USA.

[Cartwright 1989] Cartwright N. (1989). Nature's Capacities and Their Measurement. Clarendon Press, Oxford, England.

[Cartwright 1999] Cartwright N. (1999). The Dappled World. Cambridge University Press, Cambridge, England.

[Cartwright 2002] Cartwright N. (2002). Causation: One Word; Many Things*. Centre for Philosophy of Natural and Social Science, London School of Economics, Technical Report 07/03.

[Caruana & Freitag 1994] Caruana R., and Freitag D. (1994). Greedy Attribute Selection. In Proceedings of the Eleventh International Conference on Machine Learning, ML-94. Morgan Kaufmann.

[Caruana et al 1996] Caruana R., Baluja S., and Mitchell T. (1996). Using the Future to "Sort Out" the Present: Rankprop and Multitask Learning for Medical Risk Evaluation. In Advances in Neural Information Processing Systems, volume 8, pp 959–965. Morgan-Kaufmann.

[Castillo et al 2006] Castillo E., Guijarro-Berdiñas B., Fontenla-Romero O., and Alonso-Betanzos A. (2006). A Very Fast Learning Method for Neural Networks Based on Sensitivity Analysis. The Journal of Machine Learning Research, volume 7, pp. 1159-1182. MIT Press Cambridge, MA, USA.

[Castro et al 2002] Castro J. L., Mantas C. J., & Benite, J. (2002). Interpretation of artificial neural networks by means of fuzzy rules. IEEE Transactions on Neural Networks, volume 13, issue 1, pp. 101-116. IEEE.

[Cayton 2008] Cayton L. (2008). Fast Nearest Neighbor Retrieval for Bregman Divergences. Proceedings of the twenty fifth international conference on Machine learning (ICML 2008), pp 112-119. ACM New York, NY, USA.

[Chang et al 2008] Chang E., Zhu K., Wang H., Bai H., Li J., Qiu Z., and Cui H. (2008). Parallelizing Support Vector Machines on Distributed Computers. Advances in Neural Information Processing Systems 20, pp 257-264. MIT Press, Cambridge, MA, USA.

[Chapelle et al 2003] Chapelle O., Weston J., and Schölkopf B. (2003). Cluster Kernels for Semi-Supervised Learning. Advances in Neural Information Processing Systems 15, pp 585-592. MIT Press, Cambridge, MA, USA.

[Charniak & Goldman 1989] Charniak E. and Goldman R. P. (1989). Proceedings of the Fifth Annual Conference on Uncertainty in Artificial Intelligence, pp. 343-352. North-Holland Publishing Co.

[Cheesman et al 1988] Cheeseman P., Stutz J., Self M., Kelly J., Taylor W., & Freeman D. (1988). Bayesian Classification. In the Proceedings of the Seventh National Conference of Artificial Intelligence, (AAAI-88), pp. 607-611. Morgan Kaufmann Publishers, San Francisco.

[Chen, Wang & Lee 2004] Chen P-W., Wang J-Y., and Lee H-M. (2004). Model selection of SVMs using GA approach. Proceedings of the IEEE International Joint Conference on Neural Networks, volume 3, pp. 2035-2040. IEEE.

[Chen et al 2008] Chen J., Muggleton S., & Santos J. (2008). Learning probabilistic logic models from probabilistic examples. Machine Learning, volume 73, number 1, pp. 55-85. Springer Netherlands.

[Cheng et al 2002] Cheng J., Greiner R., Kelly J., Bell D., and Liu W. (2002). Learning Bayesian networks from data: An information-theory based approach. Artificial Intelligence, volume 137, issues 1-2, pp. 43-90. Elsevier Science B.V.

[Chickering et al 1994] Chickering D.M., Geiger D., and Heckerman D.E.(1994). Learning Bayesian Networks is NP-Hard. Microsoft Research Technical Report MSR-TR-94-17.

[Chickering and Heckerman 1996] Chickering D.M. and Heckerman D.E.(1996). Efficient approximations for the marginal likelihood of incomplete data given a Bayesian network. Microsoft Research Technical Report MSR-TR-96-08.

[Chopra et al 2005] Chopra S., Hadsell R., & LeCun Y. (2005). Learning a Similiarty Metric Discriminatively, with Application to Face Verification. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR-05), volume 1, pp 539- 546. San Diego, CA.

[Chow & Liu 1968] Chow C.K., & Liu C.N. (1968). Approximating Discrete Probability Distributions with Dependence Trees. IEEE Transactions on Information Theory, volume 14, pp. 462-467. IEEE press.

[Church 1940] Church A. (1940). On the Concept of a Random Sequence. Bulletin of the American Mathematical Society, volume 46, number 2, pp. 130-135.

[Churchland and Sejnowski 1992] Churchland P. S., & Sejnowski T. J. (1992). The computational brain. Cambridge, MA: The MIT Press.

[Clark & Niblett 1989] Clark P, & Niblett R. (1989). The CN2 induction algorithm. Machine Learning, 3, 261-284.

[Cohen 1994] Cohen W. (1994). Grammatically biased learning: Learning logic programs using an explicit antecedent description language. Artificial Intelligence, 68(2), pp 303-366.

[Cohen and Singer 1999] Cohen W. and Singer Y. (1999). A Simple, Fast, and Effective Rule Learner. In Proceedings of the Sixteenth National Conference on Artificial Intelligence, pp. 335-342. AAAI Press.

[Console et al 2003] Console L., Picardi C., Dupré D. T. (2003). Temporal Decision Trees: Model-based Diagnosis of Dynamic Systems On-Board. Journal of Artificial Intelligence Research, 19, pp 469-512. Available from `http://www.cs.washington.edu/research/jair/volume19/console03a.ps`.

[Cooper 1990] Cooper G. (1990). Computational complexity of probabilistic inference using Bayesian belief networks(research note). Artificial Intelligence, 42, 393-405.

[Cooper & Herskovits 1992] Cooper G., & Herskovits E. (1992). A Bayesian method for the induction of probabilistic networks from data. Machine Learning, volume 9, number 4, pp. 309-347. Springer Netherlands.

[Cooper 1997] Cooper G. (1997). A simple constraint-based algorithm for efficiently mining observational databases for causal relationships. Data Mining and Knowledge Discovery Vol 1, number 2, pp 203-224. Springer Netherlands.

[Cormen et al 1990] Cormen T.H., Leiserson C.E., and Rivest R.L. (1990). Introduction to Algorithms (1st ed.). MIT Press.

[Cortes and Vapnik 1995] Cortes C., and Vapnik V. (1995). Support-Vector Networks. Machine Learning, 20, pp 273-297.

[Cover and Hart 1967] Cover T., Hart P. (1967). Nearest neighbor pattern classification. IEEE Transactions in Information Theory, volume 13, number 1, pp 21-27. IEEE Information Theory Society.

[Craven 1996] Craven M. W. (1996). Extracting comprehensible models from trained neural networks (PhD thesis) (UW Technical Report CS-TR-96-1326). Department of Computer Sciences, University of Wisconsin-Madison.

[Craven & Shavlik 1994] Craven M. W., & Shavlik J. W. (1994). Using sampling and queries to extract rules from trained neural networks. Proceedings of the 11th International Conference on Machine Learning , pp. 37-45. San Mateo, CA: Morgan Kaufmann.

[Cybenko 1988] Cybenko G. (1988). Continuous valued neural networks with two hidden layers are sufficient (Technical Report). Department of Computer Science, Tufts University, Medford MA.

[Cybenko 1989] Cybenko G. (1989). Approximation by superpositions of a sigmoid function. Mathematics of Control, Signals, and Systems, 2, 303-314.

[da Costa & Cardoso 2005] da Costa J. P., and Cardoso J. S. (2005). Classification of Ordinal Data Using Neural Networks. Proceedings of the sixteenth European Conference on Machine Learning, ECML 2005, pp. 690-697. Springer-Verlag Berlin, Heidelberg.

[Dagum & Luby 1993] Dagum P., Luby M. (1993). Approximating probabilistic reasoning in Bayesian belief networks is NP-hard. Artificial Intelligence, volume 60, issue 1, pp. 141-153. Elsevier Science Publishers Ltd. Essex, UK.

[Darwin 1794] Darwin E. (1794). Zoönomia, or the Laws of Organic Life.

[Dasgupta & Hsu 2008] Dasgupta S., Hsu D. (2008). Hierarchical Sampling for Active Learning. Proceedings of the twenty fifth International Conference on Machine Learning, ICML 2008, pp. 208-215. ACM New York, NY, USA.

[Dayan 1993] Dayan, P. (1993). Improving generalization for temporal difference learning: The successor repre- sentation. Neural Computation, volume 5, issue 4, pp. 613–624. MIT Press Cambridge, MA, USA.

[Dayanik et al 2006] Dayanik A., Lewis D.D., Madigan D., Menkov V., and Genkin A. (2006). Constructing Informative Prior Distributions from Domain Knowledge in Text Classification. Proceedings of the Twenty Ninth annual international ACM SIGIR conference on Research and development in information retrieval, pp. 493-500. ACM New York, NY, USA.

[De Finetti 1937] De Finetti B. (1937). La Prévision: Ses Lois Logiques, Ses Sources Subjectives. Annales de l"Institut Henri Poincaré, volume 7, pp. 1-68; translated as "Foresight. Its Logical Laws, Its Subjective Sources", in Studies in Subjective Probability, editors H. E. Kyburg, Jr. and H. E. Smokler. New York: Wiley, 1964.

[De Raedt et al 2008] De Raedt L., Frasconi P., Kersting K., & Muggleton S. (2008). Probabilistic Inductive Logic Programming, Theory and Applications. In Lecture Notes in Computer Science, volume 4911. Berlin: Springer.

[DeJong 2004] DeJong G. (2004). Explanation-based learning. In Computer Science Handbook. CRC 2nd edition. Chapman & Hall.

[DeJong 2006] DeJong G. (2006). Toward Robust Real-World Inference: A New Perspective on Explanation-Based Learning. Machine Learning: ECML 2006, Proceedings of the Seventeenth European conference on machine learning, pp. 102-113. Springer Berlin.

[DeJong et al 1993] DeJong K. A, Spears W. M, & Gordon D. F. (1993). Using genetic algorithms for concept learning. Machine Learning, 13, 161-188.

[Dekel and Singer 2007] Dekel O., and Singer Y. (2007). Support Vector Machines on a Budget. Advances in Neural Information Processing Systems 19, pp 345-352. MIT Press, Cambridge, MA.

[Dempster et al 1977] Dempster A. P., Laird N. M., and Rubin D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society, Series B (Methodological), volume 39, number 1, pp. 1-38. Blackwell Publishing.

[Descartes 1644] Descartes R. (1644). Principles of Philosophy.

[Dietterich 2000] Dietterich T. (2000). An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization. Machine Learning, volume 40, pp. 139–157. Kluwer Academic Publishers, Netherlands.

[Dietterich & Kong 1995] Dietterich T. & Kong E. (1995). Machine learning bias, statistical bias, and statistical variance of decision tree algorithms. Technical Report, Department of Computer Science, Oregon State University, Corvallis, Oregon.

[Dietterich et al 1997] Dietterich T., Lathrop R., and Lozano-Pérez T. (1997). Solving the Multiple-Instance Problem with Axis-Parallel Rectangles. Artificial Intelligence, volume 89, pp 31-71. Elsevier Science Publishers B. V. Amsterdam, The Netherlands.

[Van Dijck and Van Hulle 2006] Van Dijck G. and Van Hulle M. (2006). Speeding Up the Wrapper Feature Subset Selection in Regression by Mutual Information Relevance and Redundancy Analysis. Proceeding of the sixteenth International Conference on Artificial Neural Networks, ICANN 2006, pp 31-40. Springer-Verlag Berlin Heidelberg.

[Doerr Happ & Klein 2008] Doerr B., Happ E., and Klein C. (2008). Crossover Can Provably be Useful in Evolutionary Computation. Proceedings of the tenth annual conference on Genetic and evolutionary computation, pp. 539-546. ACM New York, NY, USA.

[Domeniconi and Yan 2004] Domeniconi C. and Yan B. (2004). Nearest Neighbor Ensemble. The seventeenth International Conference on Pattern Recognition (ICPR'04) - Volume 1, pp 228-231. IEEE Press.

[Domingos & Pazzani 1997] Domingos, P., & Pazzani, M. (1997). On the optimality of the simple Bayesian classifier under zero-one loss. Machine Learning, volume 29, pp. 103-130. Kluwer Academic Publishers Hingham, MA, USA.

[Dougherty et al 1995] Dougherty J., Kohavi R., and Sahami M. (1995). Supervised and Unsupervised Discretization of Continuous Features. In Proceedings of the Twelfth International Conference on Machine Learning, pp. 194-202. Morgan Kaufmann.

[Dowe 2000] Dowe P. (2000). Physical causation. Cambridge University Press.

[Downing 2001] Downing K.L. (2001). Reinforced Genetic Programming. Genetic Programming and Evolvable Machines, volume 2, number 3, pp. 259-288. Springer Netherlands.

[Dretske 1981] Dretske F. (1981). Knowledge and the Flow of Information. MIT Press, Cambridge, Massachusetts, USA.

[Duda & Hart 1973] Duda R. O. & Hart P. E. (1973). Pattern Classification and Scene Analysis. New York: John Wiley & Sons.

[Dudani 1976] Dudani, S. (1976). The distance-weighted k-nearest neighbor rule. IEEE Transactions on Systems, Man and Cybernetics, volume 6(4), pp 325–327.

[Dudley 2002] Dudley R.M. (2002). Real Analysis and Probability. Cambridge University Press.

[Dugas et al 2009] Dugas C., and Bengio Y., and Bélisle F., and Nadeau C., and Garcia R. (2009). Incorporating Functional Knowledge in Neural Networks. Journal of Machine Learning Research, volume 10, pp. 1239-1262. MIT Press, Cambridge, MA, USA.

[Dutch et al 2001] Duch W., Adamczak R., & Grabczekski K. (2001). A new methodology of extraction, optimization and application of crisp and fuzzy logical rules. IEEE Transactions on Neural Networks, volume 12, issue 2, pp. 277–306. IEEE.

[Dutch et al 2004] Duch, W., Setiono R., and Zurada J. (2004). Computational Intelligence Methods for Rule-Based Data Understanding. Proceedings of the IEEE, volume 92, issue 5, pp. 771-805. IEEE.

[Efron et al 2004] Efron B., Hastie T., Johnstone L. Tibshirani R. (2004). Least Angle Regression. Annals of Statistics, volume 32, pp 407-499. American Mathematical Society, 201 Charles Street, Providence, RI 02904-6248 USA.

[Efron & Tibshirani 1994] Efron B., and Tibshirani R. (1994). Introduction to the Bootstrap (Chapman & Hall/CRC Monographs on Statistics & Applied Probability). Chapman & Hall, CRC Press, USA.

[Eitrich and Lang 2006] Eitrich T. and Lang B. (2006). Efficient optimization of support vector machine learning parameters for unbalanced datasets. Journal of Computational and Applied Mathematics, Volume 196 , Issue 2, pp 425-436. Elsevier Science Publishers B. V. Amsterdam, The Netherlands.

[Elga 2000] Elga A., (2000). Self-Locating Belief and the Sleeping Beauty Problem. Analysis, volume 60, issue 2, pp. 143-147. Oxford University Press.

[Elman 1990] Elman J. L. (1990). Finding structure in time. Cognitive Science, 14, 179-211.

[Eriksson & Hájek 2007] Eriksson L. and Hájek A. (2007). What Are Degrees of Belief? Studia Logica, volume 86, number 2, pp. 185-215. Editor Branden Fitelson. Springer.

[Facione 1977] Facione P.A. (1977). The Entailment Operator. Notre Dame Journal of Formal Logic, volume 18, number 3, pp. 415-420. Duke University Press.

[Fahlam & Lebiere 1990] Fahlam S, & Lebiere C. (1990). The CASCADE-CORRELATION learning architecture (Technical Report CMU-CS-90-100). Computer Science Department, Carnegie Mellon University, Pittsburgh, PA.

[Fenton & Wang 2006] Fenton N.E. and Wang W. (2006). Risk and Confidence Analysis for Fuzzy Multicriteria Decision Making. Knowledge Based Systems, volume 19, pp. 430-437. Elsevier Science Publishers B. V. Amsterdam, The Netherlands.

[Fenton and Neil 2000] Fenton N. E. and Neil M. (2000). The "Jury Observation Fallacy" and the use of Bayesian Networks to present Probabilistic Legal Arguments, Mathematics Today (Bulletin of the IMA, 36(6)), 180-187.

[Fine 1973] Fine T. L. (1973). Theories of Probability: An Examination of Foundations. Academic Press.

[Fix and Hodges 1951] Fix, E., Hodges, J. (1951). Discriminatory analysis, nonparametric discrimination: Consistency properties. Technical Report 4, USAF School of Aviation Medicine, Randolph Field, Texas.

[Flores et al 2009] Flores M. J., Gámez J. A., Martínez A. M. and Puerta J. M. (2009). GAODE and HAODE: Two Proposals based on AODE to Deal with Continuous Variables. Proceedings of the Twenty Sixth Annual International Conference on Machine Learning, pp. 313-320. ACM New York, NY, USA.

[Frank & Bouckaert 2006] Frank E. and Bouckaert R. R. (2006). Naive Bayes for Text Classification with Unbalanced Classes. Knowledge Discovery in Databases: PKDD 2006, pp. 503-510. Springer, Berlin.

[Freedman & Humphreys 1999] Freedman D. A. and Humphreys P. (1999). Are There Algorithms That Discover Causal Structure? Synthese, volume 121, numbers 1-2, pp 29-54. Springer Netherlands.

[Freund 1999] Freund Y. (1999). An adaptive version of the boost by majority algorithm. Proceedings of the Twelfth Annual Conference on Computational Learning, pp 102-113. ACM New York, NY, USA.

[Freund and Schapire 1995] Freund Y., and Schapire R. (1995). A decision-theoretic generalization of on-line learning and an application to boosting. Proceedings of the Second European Conference on Computational Learning Theory, pp 23-37. Springer-Verlag London, UK.

[Freidman et al 1977] Freidman J., Bentley J., Finkel R. (1977). An Algorithm for Finding Best Matches in Logarithmic Expected Time. ACM Transactions on Mathematical Software, volume 3, 3, pp 209-226. ACM Press.

[Friedman & Goldszmidth 1996] Friedman N. and Goldszmidt M. (1996). Building Classifiers using Bayesian Networks. In Proceedings of the Thirteenth National Conference on Artificial Intelligence, pp. 1277-1284. AAAI Press.

[Frome et al 2007] Frome A., and Singer Y., and Sha F., and Malik J. (2007). Learning Globally-Consistent Local Distance Functions for Shape-Based Image Retrieval and Classification. Proceedings of the Eleventh IEEE International Conference on Computer Vision (ICCV-07), pp 1-8.

[Fu 1993] Fu L. M. (1993). Knowledge-based connectionism for revising domain theories. IEEE Transactions on Systems, Man, and Cybernetics, 23(1), 173-182.

[Fürnkranz 1999] Fürnkranz J. (1999). Separate-and-Conquer Rule Learning. Artificial Intelligence Review, volume 13, pp. 3-54. Springer Netherlands.

[Fürnkranz & Widmer 1994] Fürnkranz J. and Widmer G. (1994). Incremental Reduced Error Pruning. Proceedings of the Eleventh International Conference on Machine Learning (ICML 1994), pp. 70-77. Morgan Kaufmann.

[Van Der Gaag & Coupé 1999] Van Der Gaag L.C., and Coupé V.M.H. (1999). Sensitivity Analysis for Threshold Decision Making with Bayesian Belief Networks. Advances in Artificial Intelligence, Lecture Notes in Artificial Intelligence, pp. 37-48. Springer-Verlag.

[Gabriel and Moore 1990] Gabriel M. & Moore J. (1990). Learning and computational neuroscience: Foundations of adaptive networks (edited collection). Cambridge, MA: The MIT Press.

[Geiger & Heckerman 1995] Geiger D. and Heckerman D. (1995). A characterization of the Dirichlet distribution applicable to learning Bayesian networks. Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence, pp. 196-207. Morgan Kaufmann.

[Gelfand & Dey 1994] Gelfand A.E., and Dey D.K. (1994). Bayesian Model Choice: Asymptotics and Exact Calculations. Journal of the Royal Statistical Society, Series B, volume 56, pp. 501-514. Royal Statistical Society.

[Gentner 1983] Gentner D. (1983). Structure-mapping: A Theoretical Framework for Analogy. Cognitive Science, volume 7, issue 2, pp. 155-170. Elsevier Inc.

[Getoor & Taskar 2007] Getoor L., & Taskar B. (2007). Adaptive Computation and Machine Learning. Introduction to statistical relational learning. Cambridge: MIT Press.

[Gettier 1963] Gettier E.L. (1963). Is Justified True Belief Knowledge? Analysis volume 23, number 6, pp. 121-123. Oxford University Press, England.

[Ghahramani 1998] Ghahramani Zoubin. (1998). Learning Dynamic Bayesian Networks. Adaptive Processing of Sequences and Data Structures. Lecture Notes in Computer Science 1387, pp. 168-197. Springer-Verlag, Berlin.

[Giere 1973] Giere R. N. (1973). Objective Single-Case Probabilities and the Foundations of Statistics. Proceedings of the Fourth International Congress for Logic, Methodology and Philosophy of Science, pp. 467-483. North-Holland Publishing Company.

[Gillies 2000] Gillies D. (2000). Varieties of Propensity. British Journal for the Philosophy of Science, volume 51, issue 4, pp. 807-835. Oxford University Press.

[Ginsberg 1972] Ginsberg M. (1972). The Entailment-Presupposition Relationship. Notre Dame Journal of Formal Logic, volume XIII, number 4, pp. 511-515. Duke University Press.

[Giraud-Carrier 2000] Giraud-Carrier C. (2000). Unifying Learning with Evolution Through Baldwinian Evolution and Lamarckism: A Case Study. Technical Report: CS-EXT-2000-308, University of Bristol, Bristol, UK.

[Glymour et al 1987] Glymour C., Scheines R., Spirtes P. and Kelly K. (1987). Discovering Causal Structure: Artificial Intelligence, Philosophy of Science, and Statistical Modeling. Academic Press, San Diego, CA.

[Goldberg 1989] Goldberg D. (1989). Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley.

[Golea et al 1998] Golea M., Bartlett P., Lee W., and Mason L. (1998). Generalization in decision trees and DNF: does size matter? Proceedings of the 1997 conference on Advances in neural information processing systems 10, pp 259-265. MIT Press Cambridge, MA, USA.

[Good 1950] Good I. J. (1950). Probability and the weighing of evidence. Griffin, London.

[Good 1961] Good I.J. (1961). A Causal Calculus. The British Journal for the Philosophy of Science, volume 44, pp. 305-318. Oxford University Press.

[Goodman 1955] Goodman H. N. (1955). Fact, Fiction, and Forecast. Harvard University Press.

[Goodman et al 2004] Goodman J., O'Rourke J. and Indyk P. (2004). Chapter 39: Nearest neighbors in high-dimensional spaces. Handbook of Discrete and Computational Geometry (2nd ed.). CRC Press.

[Gorissen et al 2009] Gorissen D., Dhaene T. and De Turck F. (2009). Evolutionary Model Type Selection for Global Surrogate Modeling. Journal of Machine Learning Research, volume 10, pp. 2039-2078. Microtome Publishing.

[Graf et al 2005] Graf H.P., Cosatto E., Bottou L., Dourdanovic I., and Vapnik V. (2005). Parallel Support Vector Machines: The Cascade SVM. Advances in Neural Information Processing Systems 17, pp 521-528. MIT Press Cambridge, MA, USA.

[Grandvalet et al 2009] Grandvalet Y., Rakotomamonjy A., Keshet J. and Stephane Canu. (2009). Support Vector Machines with a Reject Option. Advances in Neural Information Processing Systems 21, pp 537-544.

[Granger 1969] Granger C.W.J. (1969). Investigating Causal Relations by Econometric Models and Cross-spectral Methods. Econometrica, volume 37, number 3, pp. 424-438. The Econometric Society.

[Grenfenstette 1991] Grefenstette J.J. (1991). Lamarckian learning in multi-agent environments. In R. Belew and L. Booker (Eds.), Proceedings of the Fourth International Conference on Genetic Algorithms. San Mateo, CA: Morgan Kaufmann.

[Guyon and Elisseeff 2003] Guyon I. and Elisseeff A. (2003). An introduction to variable and feature selection. Journal of Machine Learning Research, volume 3, pp 1157–1182. MIT Press.

[Haavelmo 1943] Haavelmo T. (1943). The statistical implications of a system of simultaneous equations. Econometrica, volume 11, number 1, pp. 1–12. The Econometric Society. JSTOR URL `http://www.jstor.org/stable/1905714`.

[Hachiya et al 2009] Hachiya H., Peters J. and Sugiyama M. (2009). Efficient Sample Reuse in EM-based Policy Search. Machine Learning and Knowledge Discovery in Databases, volume 5781, pp. 469-484. Springer, Berlin.

[Hájek 2010] Hájek A. (2010). Interpretations of Probability, The Stanford Encyclopedia of Philosophy. URL `http://plato.stanford.edu/archives/spr2010/entries/probability-interpret/`.

[Halmos 1974] Halmos P.R. (1974). Measure Theory. Springer-Verlag.

[Halpern & Pearl 2000] Halpern J.Y., Pearl J. (2000). Causes and Explanations: A Structural-Model Approach, Part I: Causes. The British Journal for the Philosophy of Science volume 56, issue 4, pp. 843-887. Morgan Kaufmann.

[Hamming 1950] Hamming, R.W. (1950). Error detecting and error correcting codes. Bell System Technical Journal, volume 26, number 2, pp. 147-160.

[Hampshire & Waibel 1989] Hampshire J. & Waibel A. (1989). The meta-pi network: Building distributed knowledge representations for robust pattern recognition. Carnegie Mellon University technical report CMU-CS-89-166.

[Hart et al 1978] Hart P. E., Duda R. O. and Einaudi M. T. (1978). PROSPECTOR—A computer-based consultation system for mineral exploration. Mathematical Geology, volume 10, number 5, pp. 589-610. Springer.

[Hashem and Cooper] Hashem A. I. and Cooper G. F. (1996). Human Causal Discovery from Observational Data. Proceedings of the American Medical Informatics Association, AMIA, annual symposium 1996, pp. 27-31. AMIA CD-ROM only. Available from `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2233172/pdf/procamiaafs00002-0064.pdf`.

[Heckerman 1995a] Heckerman D. (1995). A Bayesian Approach to Learning Causal Networks. Microsoft Research technical report, MSR-TR-95-04. Morgan Kaufmann.

[Heckerman 1995b] Heckerman D. (1995). A Tutorial on Learning With Bayesian Networks. Microsoft Technical Report, MSR-TR-95-06. Microsoft Research.

[Heckerman & Shachter 1994] Heckerman D., and Shachter R. (1994). A decision-based view of causality. Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence, pp 302-310. Morgan Kaufmann.

[Heckerman et al 1995] Heckerman D., Geiger D., Chickering D.M. (1995). Learning Bayesian Networks: The Combination of Knowledge and Statistical Data. Machine Learning 20, pp 197-243. Springer Netherlands.

[Heckerman et al 1997] Heckerman D., Meek C., Cooper G. (1997). A Bayesian Approach to Causal Discovery. Microsoft Technical Report, MSR-TR-97-05. Microsoft Research.

[Heiler et al 2001] Heiler M., Cremers D., Schnörr C. (2001). Efficient Feature Subset Selection for Support Vector Machines. Technical Report 21/2001 Computer Science Series, Department of Mathematics and Computer Science, University of Mannheim.

[Henderson 1954] Henderson G.P. (1954). Causal Implication. Mind, New Series, volume 63, number 252, pp. 504-518. Oxford University Press.

[Herskovits & Cooper 1990] Herskovits E. & Cooper G.F. (1990). Kutató: An Entropy-Driven System for Construction of Probabilistic Expert Systems from Databases. Proceedings of the Sixth Annual Conference on Uncertainty in Artificial Intelligence, pp. 117-128. Elsevier Science Inc. New York, NY, USA.

[Hinkle & Toomey 1995] Hinkle D. and Toomey C. (1995). Applying Case-Based Reasoning To Manufacturing. AI Magazine, volume16, number 1, pp. 65-73. AAAI press.

[Hinton et al 2006] Hinton G. E., Osindero S., and Teh Y. W. (2006). A fast learning algorithm for deep belief nets. Neural Computation, volume 18, pp. 1527-1554. MIT Press, Cambridge, MA.

[Hitchcock 2004] Hitchcock C. (2004). Probability and Chance: Philosophical Aspects. The International Encyclopedia of the Social and Behavioral Sciences, volume 18, pp. 12089-12095. London: Elsevier.

[Hoi and Jin 2008] Hoi S. and Jin R. (2008). Active kernel learning. Proceedings of the twenty fifth international conference on Machine learning (ICML 2008), pp 400-407. ACM New York, NY, USA.

[Holland 1986] Holland J.H. (1986). Escaping brittleness: The possibilities of general-purpose learning algorithms applied to parallel rule-based systems. In R. Michalski, J. Carbonell, & T Mitchell (Eds.), Machine Learning: An artificial intelligence approach (Vol 2). San Mateo, CA: Morgan Kaufmann.

[Hornick et al 1989] Hornick K, Stinchcombe M, & White H. (1989). Multilayer feedforward networks are universal approximators. Neural Networks, 2, 359-366.

[Hsu, Huang & Wong 2000] Hsu C.-N., Huang H.-J., and Wong T.-T. (2000). Why Discretization Works for Naive Bayesian Classifiers. Proceedings of the Seventeenth International Conference on Machine Learning, pp. 399-406. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA.

[Hume 1748] Hume D. (1748). An Enquiry concerning Human Understanding.

[Humphreys 1985] Humphreys P. (1985). Why Propensities Cannot be Probabilities. The Philosophical Review, volume 94, number 4, pp. 557-570. Duke University Press.

[Igel & Husken 2003] Igel C., and Husken M. (2003). Empirical Evaluation of the Improved Rprop Learning Algorithms. Neurocomputing, volume 50, pp. 105-123. Elsevier.

[Jacobs et al 1991a] Jacobs R.A., Jordan M. I., & Barto A. G. (1991). Task Decomposition Through Competition in a Modular Connectionist Architecture. Cognitive Science, volume 15, pp. 219-250. Cognitive Science Society.

[Jacobs et al 1991b] Jacobs R.A., Jordan M.I., Nowlan S.J., and Hinton G.E. (1991). Adaptative Mixtures of Local Experts. Neural Computation volume 3, number 1, pp. 79-87. MIT Press.

[Jagadeesh et al 2008] Jagadeesh R. P., Bose C., Nagaraja G. (2008). Performance Studies on KBANN. Fourth International Conference on Hybrid Intelligent Systems (HIS'04), pp. 198-203. IEEE.

[James 1911] James W. (1911). The Meaning of Truth. Longman Green and Co, New York, USA.

[Janikow 1993] Janikow C. Z. (1993). A knowledge-intensive GA for supervised learning. Machine Learning, 13, 189-228.

[Jaynes 1968] Jaynes E. T. (1968). Prior Probabilities. Institute of Electrical and Electronic Engineers Transactions on Systems Science and Cybernetics, volume 4, issue 3, pp. 227-241. IEEE press.

[Jaynes 1973] Jaynes E. T. (1973). The Well-Posed Problem. Foundations of Physics, volume 3, pp. 477-493. Plenum Publishing Corporation.

[Jeffrey 1965] Jeffrey R. (1965). The Logic of Decision. The University of Chicago Press, Chicago, USA.

[Jevons 1874] Jevons W. S.(1874). The principles of science: A treatise on logic and scientific method. London: Macmillan.

[Joachims 2002] Joachims S. (2002). Learning to Classify Text using Support Vector Machines: Methods, Theory, and Algorithms. Kluwer Academic Publishers / Springer.

[John & Langley 1995] John G. H. and Langley P. (1995). Estimating Continuous Distributions in Bayesian Classifiers. Proceedings of the Eleventh Annual Conference on Uncerntainty in Artificial Intelligence, pp 338-345. Morgan Kaufmann.

[John 1997] John G. H. (1997). Enhancements to the Data Mining Process. PhD thesis, Stanford University.

[Johnson 1992] Johnson L.E. (1992). Focusing on Truth. Routledge, London, England.

[Jorden 1986] Jorden M. (1986). Attractor dynamics and parallelism in a connectionist sequential machine. Proceedings of the Eighth Annual Conference of the Cognitive Science Society, pp. 531-546.

[Jorden & Jacobs 1993] Jordan M.I. and Jacobs R.A. (1993). Hierarchical Mixtures of Experts and the EM Algorithm. Proceedings of 1993 International Joint Conference on Neural Networks, volume 2, pp. 1339-1344. IEEE press.

[Jugelenaite & Heskes 2006] Jugelenaite R. and Heskes T. (2006). EM Algorithm for Symmetric Causal Independence Models. Machine Learning: ECML 2006, pp. 234-245. Springer, Berlin.

[Kääriäinen et al 2004] Kääriäinen M., Malinen T., and Elomaa T.(2004). Selective Rademacher Penalization and Reduced Error Pruning of Decision Trees. The Journal of Machine Learning Research, volume 5, pp. 1107-1126. MIT Press Cambridge, MA, USA.

[Kant 1781] Kant I. (1781). Kritik der reinen Vernunft. Critique of pure reason by Immanuel Kant : translated by Norman Kemp Smith Macmillan, London : 1929.

[Karimi and Hamilton 2002] Karimi K. and Hamilton H. J.(2002). RFCT: An Association-Based Causality Miner. The Fifteenth Canadian Conference on Artificial Intelligence, (AI2002) pp 334-338. Springer-Verlag Berlin Heidelberg.

[Kearns 1988] Kearns M.(1988). Thoughts on hypothesis boosting, ML class project (unpublished).

[Kearns & Mansour 1998] Kearns M., & Mansour Y. (1998). A Fast, Bottom-Up Decision Tree Pruning Algorithm with Near-Optimal Generalization. Proceedings of the Fifteenth International Conference on Machine Learning, pp 269-277. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA.

[Kedar-Cabelli & McCarty 1987] Kedar-Cabelli S., & McCarty T. (1987). Explanation-based generalization as resolution theorem proving. Proceedings of the Fourth International Workshop on Machine Learning (pp. 383-389). San Francisco: Morgan Kaufnamm.

[Kemeny 1955] Kemeny J. (1955). Fair Bets and Inductive Probabilities. Journal of Symbolic Logic, volume 20, issue 3, pp. 263-273. Association for Symbolic Logic.

[Keogh & Pazzani 1999] Keogh E. J., & Pazzani M. J. (1999). Learning augmented Bayesian classifiers: A comparison of distribution-based and classification-based approaches. In Proceedings of the Seventh International Workshop on Artificial Intelligence and Statistics, pp. 225–230. Society for Artificial Intelligence and Statistics.

[Keynes 1921] Keynes J. M. (1921). A Treatise on Probability. Macmillan and Co.

[Keynes 1939] Keynes J. M.(1939). Professor Tinbergen's Method, The Economics Journal 49, pp 558-70.

[Keynes 1940] Keynes J. M.(1940). Comments on Tinbergen's Response. The Economics Journal 50, pp 154-56.

[Kim et al 2006] Kim S.-B., Han K.-S., Rim H.-C., and Myaeng S. H. (2006). Some Effective Techniques for Naive Bayes Text Classification. IEEE Transactions on Knowledge and Data Engineering, volume 18, number 11, pp. 1457-1466. IEEE Press.

[Kimming et al 2007] Kimmig A., Raedt L., and Toivonen H. (2007). Probabilistic Explanation Based Learning. Proceedings of the Eighteenth European conference on Machine Learning, ECML 2007, pp. 176-187. Springer-Verlag Berlin.

[Kitano et al 1992] Kitano H., Shibata A., Shimazu H., Kajihara J., & Sato A. (1992). Building large-scale and corporate-wide case-based systems: integration of organizational and machine executable algorithms. In Proceedings of the Tenth National Conference on Artificial Intelligence, pp. 843-849. AAAI Press.

[Kittler 1978] Kittler J.(1978). Feature set search algorithms, Pattern Recognition and Signal Processing, C. H. Chen, Ed. Amsterdam: Sijtho & Noordho.

[Kjærulff 1992] Kjærulff U. (1992). A Computational Scheme for Reasoning in Dynamic Probabilistic Networks. Proceedings of the Eighth Annual Conference on Uncertainty in Artificial Intelligence, UAI 92, pp. 121-129. Morgan Kaufmann.

[Klopotek 1994] Klopotek M.A. (1994). Learning Belief Network Structure From Data under Causal Insufficiency. Proceedings of the European conference on machine learning on Machine Learning 1994, ECML-94, pp. 379-382. Springer-Verlag New York, Inc. Secaucus, NJ, USA.

[Klopotek 2000] Klopotek M.A. (2000). On a Deficiency of the FCI Algorithm Learning Bayesian Networks from Data. Demonstratio Mathematica, volume XXXIII, number 1, pp. 181-194. Warsaw University of Technology, Warsaw Poland.

[Kohavi 1995] Kohavi R.(1995). A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. Proceedings of the International Joint Conference on Artificial Intelligence, (IJCAI 1995), pp 1137-1143. Morgan Kaufmann.

[Kohavi and John 1997] Kohavi R. and John G. H.(1997). Wrappers for Feature Subset Selection. Artificial Intelligence, Vol. 97, No. 1-2, pp 273-324. Elsevier Science Publishers B. V. Amsterdam, The Netherlands.

[Kohavi & Sahami 1996] Kohavi R. and Sahami M. (1996). In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, pp. 114-119. AAAI Press.

[Kohavi & Sommerfield 1995] Kohavi Ron and Sommerfield Dan.(1995). Feature subset selection using the wrapper method: Overfitting and dynamic search space, pp 192-197. AAAI Press.

[Koller & Sahami 1996] Koller D., & Sahami M.(1996). Toward Optimal Feature Selection. In: Machine Learning: Proceedings of the Thirteenth International Conference. Morgan Kaufmann.

[Kolmogorov 1933] Kolmogorov A. (1933). Grundbegriffe der Wahrscheinlichkeitsrechnung. Berlin: Julius Springer. Translation: Kolmogorov A. (1956). Foundations of the Theory of Probability. New York: Chelsea Publishing Company.

[Kolodner 1983] Kolodner J. L. (1983). Reconstructive memory: A computer model. Cognitive Science volume 7, issue 4, pp. 281-328. Elsevier Inc.

[Kondor and Lafferty 2002] Kondor R. I. and Lafferty J.(2002). Diffusion Kernels on Graphs and Other Discrete Structures. Proceedings of the nineteenth International Conference on Machine Learning (ICML 2002), pp 315-322. Morgan Kaufmann.

[Korb 1999] Korb K. B.(1999). Probabilistic causal structure, in H. Sankey (ed.); Causation and laws of nature, Dordrechet: Kluwer, pp 265-311.

[Koton 1988] Koton P. (1988). Using experience in learning and problem solving. Massachusetts Institute of Technology, Laboratory of Computer Science, Ph.D. dissertation, MIT/LCS/TR-441.

[Koza 1992] Koza J.(1992). Genetic programming: On the programming of computers by means of natural selection. Cambridge, MA: MIT Press.

[Koza et al 1996] Koza J. R., Bennet III F. H, Andre D, & Keane M. A.(1996). Four problems for which a computer program evolved by genetic programming is competitive with human performance. Proceedings of the 1996 International Conference on Evolutionary Computation (pp. 1-10). IEEE Press.

[Kripke 1959] Kripke S. A. (1959). A Completeness Theorem in Modal Logic. The Journal of Symbolic Logic volume 24, number 1, pp. 1-14. Association for Symbolic Logic.

[Kripke 1963] Kripke S. A. (1963). Semantical Considerations on Modal Logic. Acta Philosophica Fennica, volume 16, pp. 83-94. Philosophical Society of Finland.

[Kulis et al 2006] Kulis B., Sustik M., & Dhillon I.(2006). Learning low-rank kernel matrices. Proceedings of the twenty third international conference on Machine learning (ICML 2006), pp 505-512. ACM New York, NY, USA.

[Kullback & Leibler 1951] Kullback S. and Leibler R.A. (1951). On Information and Sufficiency. The Annals of Mathematical Statistics, volume 22, number 1, pp. 79-86. Institute of Mathematical Statistics.

[Kwoh & Gillies 1996] Kwoh C.K. and Gillies D.F. (1996). Using hidden nodes in Bayesian networks. Artificial Intelligence, volume 88, issues 1-2, pp. 1-38. Elsevier Science B.V.

[Lam & Bacchus 1994] Lam W. and Bacchus F. (1994). Learning Bayesian Belief Networks: An approach based on the MDL Principle. Computational Intelligence, volume 10, issue 3, pp. 269-293. John Wiley & Sons, Inc.

[Lamarck 1809] Lamarck J.-B. (1809). Philosophie Zoologique.

[Lanckriet et al 2004] Lanckriet G., Cristianini N., Bartlett P., Ghaoui L. E., Jordan M. (2004). Learning the Kernel Matrix with Semidefinite Programming. The Journal of Machine Learning Research, Volume 5, pp 27-72. MIT Press Cambridge, MA, USA.

[Landwehr et al 2007] Landwehr N., Kersting K., and De Raedt L. (2007). Integrating Naive Bayes and FOIL. The Journal of Machine Learning Research, Volume 8, pp. 481-507. Microtome Publishing.

[Langley et al 1992] Langley P., Iba W., & Thompson K.(1992). An analysis of Bayesian classifiers. Proceedings of the Tenth National Conference on Artificial Intelligence, pp. 223-228. San Jose, CA: AAAI Press.

[Langseth & Nielsen 2005] Langseth H. and Nielsen T. D. (2005). Latent Classification Models. Machine Learning, volume 59, pp. 237–265. Springer, Netherlands.

[Laplace 1814] Laplace P.S. (1814, english edition 1951). A Philosophical Essay on Probabilities. New York: Dover Publications Inc.

[Larochelle et al 2009] Larochelle H., Bengio Y., Louradour J., Lamblin P. (2009). Exploring Strategies for Training Deep Neural Networks. Journal of Machine Learning Research, volume 10, pp. 1-40. Microtome Publishing.

[Larranaga et al 1996] Larranaga P., M. Poza, Yurramendi Y., Murga R. H., Kuijpers C. M. H.(1996). Structure Learning of Bayesian Networks by Genetic Algorithms: A Performance Analysis of Control Parameters. IEEE Journal on Pattern Analysis and Machine Intelligence 18(9): 912-926.

[Lauer and Bloch 2007] Lauer F. and Bloch G.(2007). Incorporating prior knowledge in support vector regression. Machine Learning, Volume 70, Number 1, pp 89-118. Springer Netherlands.

[Law and Zaniolo 2005] Law Y.-N. and Zaniolo C.(2005). An Adaptive Nearest Neighbor Classification Algorithm for Data Streams. Knowledge Discovery in Databases: PKDD 2005, pp 108-120. Springer, Berlin, Germany.

[LeCun et al 1998] LeCun Y., Bottou L., Orr G. B., and Müller K.-R. (1998). Efficient Backprop. Neural Networks, Tricks of the Trade, Lecture Notes in Computer Science LNCS 1524, Springer Verlag.

[LeCun et al 1990] LeCun Y, Denker J. S, Solla S. A.(1990). Optimal brain damage. In D. Touretzky (Ed.), Advances in Neural Information Processing Systems (Vol. 2, pp. 589-605). San Mateo, CA: Morgan Kaufmann.

[LeCun & Bengio 2007] LeCun Y. and Bengio Y. (2007). Scaling Learning Algorithms Towards AI. In Large-Scale Kernel Machines. MIT Press.

[Lee and Wong 1977] Lee D. and Wong, C.(1977). Worst-case analysis for region and partial region searches in multidimensional binary search trees and balanced quad trees. Acta Informatica volume 9 (1), pp. 23-29. Springer Berlin, Germany.

[Lee 2003] Lee H. K. H. (2003). A Noninformative Prior for Neural Networks. Machine Learning, volume 50, numbers 1-2, pp. 197-212. Springer Netherlands.

[Leibniz 1710] Leibniz G. (1710). Essais de Théodicée sur la bonté de Dieu, la liberté de l'homme et l'origine du mal (Essays on the Goodness of God, the Freedom of Man and the Origin of Evil).

[Lelewer & Hirschberg 1987] Lelewer D.A. and Hirschberg D.S. (1987). Data compression. ACM Computing Surveys (CSUR), volume 19 , issue 3, pp. 261-296. ACM New York, NY, USA.

[Lenat & Feigenbaum 1991] Lenat D. B. and Feigenbaum E. A. (1991). On the thresholds of knowledge. Artificial Intelligence, volume 47, issue 1, number 3, pp. 185-250. Elsevier Science Publishers Ltd.

[Lewis & Langford 1932] Lewis, C.I. and Langford C.H. (1932). Symbolic Logic . New York: Century Company.

[Lewis 1973a] Lewis D. (1973). Causation. The Journal of Philosophy, volume 70, number 17, pp. 556-567. Journal of Philosophy, Inc.

[Lewis 1973] Lewis D. K. (1973). Causation, with poscripts in [Lewis 1986b] pp 159-213.

[Lewis 1979] Lewis D. (1979). Attitudes De Dicto and De Se. Philosophical Review, volume 88, number 4, pp. 513–543. Duke University Press.

[Lewis 1986a] Lewis D. (1986). Probabilities of Conditionals and Conditional Probabilities II. The Philosophical Review, volume 95, number 4, pp. 581-589. Duke University Press.

[Lewis 1986b] Lewis D. K. (1986). Philosophical papers volume II, Oxford: Oxford University Press.

[Lewis 1986c] Lewis D. K. (1986). On the Plurality of Worlds. Blackwell, Oxford, England.

[Lewis 1995] Lewis D. (1995). Reduction of Mind, in A Companion to the Philosophy of Mind. Editor S. Guttenplan. Oxford: Blackwell Publishing.

[Lewis 2000] Lewis D. (2000). Causation as Influence. The Journal of Philosophy, volume 97, number 4, Special Issue: Causation, pp. 182-197. Journal of Philosophy, Inc.

[Lewis 1959] Lewis P.M. (1959). Approximating probability distributions to reduce storage requirement. Information and Control, volume 2, pp. 214-225. Elsevier B.V.

[Lin et al 2005] Lin C. T., Yeh C. M., Chung J. F., Liang S. F. and Pu H. C. (2005). Support-Vector-Based Fuzzy Neural Networks. International Journal of Computational Intelligence Research, Vol.1, No.2, pp. 138-150. Research India Publications, Delhi India.

[Ling & Zhang 2002] Ling C. X., and Zhang H. (2002). The Representational Power of Discrete Bayesian Networks. Journal of Machine Learning Research, volume 3, pp. 709-721. MIT Press Cambridge, MA, USA.

[Long and Servedio 2008] Long P. and Servedio R. (2008). Random Classification Noise Defeats All Convex Potential Boosters. Proceedings of the International Conference on Machine Learning 2008, (ICML 2008), pp 608-615. ACM Press.

[López & Plaza 1993] López B. and Plaza E. (1993). Case-Based Planning for Medical Diagnosis. Proceedings of the 7th International Symposium on Methodologies for Intelligent Systems, pp. 96-105. Springer Verlag, London.

[Lopez 1991] Lopez de Mantaras R. (1991). A distance based attribute selection measure for decision tree induction. Machine Learning, volume 6, issue 1, 81-92. Springer Netherlands.

[Luo 2006] Luo W. (2006). Learning Bayesian Networks in Semi-deterministic Systems. Lecture Notes in Computer Science, volume 4013, pp. 230-241. Springer, Berlin.

[MacKay 1992] MacKay D. J. C. (1992). Bayesian methods for adaptive methods. Ph.D. thesis, California Institute of Technology, Program in Computation and Neural Systems.

[Madigan et al 1995] Madigan D., Garvin J., and Raftery A. (1995). Eliciting prior information to enhance the predictive performance of Bayesian graphical models. Communications in Statistics: Theory and Methods, volume 24, pp. 2271-2292. Taylor & Francis, Philadelphia, USA.

[Makino 2009] Makino T. (2009). Proto-Predictive Representation of States with Simple Recurrent Temporal-Difference Networks . Proceedings of the 26th Annual International Conference on Machine Learning, pp. 697-704. ACM New York, NY, USA.

[Malerba et al 1995] Malerba D, Floriana E, & Semeraro G.(1995). A further comparison of simplification methods for decision tree induction. In D. Fisher & H. Lenz (Eds.), Learning from data: AI and Statistics. Springer-Verlag.

[Manago et al 1993] Manago M., Althoff K.-D., Auriol E., Traphöner R., Wess S., Conruyt N., and Maurer F. (1993). Induction and reasoning from cases. Proceedings of the First European Workshop on Case-Based Reasoning, pp. 313-318. Springer-Verlag.

[Mangasarian et al 2004] Mangasarian O., Shavlik J., Wild E.(2004). Knowledge-Based Kernel Approximation. The Journal of Machine Learning Research Volume 5, pp 1127-1141. MIT Press Cambridge, MA, USA.

[Mangasarian and Wild 2007] Mangasarian O., & Wild E. (2007). Nonlinear knowledge in kernel approximation. IEEE Transactions on Neural Networks, volume 18, pp. 300-306. IEEE Press.

[Mani, Spirtes & Cooper 2006] Mani S., Spirtes P., Cooper G.F. (2006). A theoretical study of Y structures for causal discovery. Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI-2006) pp. 314-323. AUAI Press.

[Mansour & McAllester 2000] Mansour Y., and McAllester D. (2000). Generalization Bounds for Decision Trees. In Proceedings of the Thirteenth Annual Conference on Computational Learning theory, (COLT 2000), pp. 69-74. Morgan Kaufmann Publishers, San Francisco, CA.

[Margaritis & Thrun 2000] Margaritis D. and Thrun S. (2000). Bayesian network induction via local neighborhoods. Advances in Neural Information Processing Systems, volume 12, pp. 505–511. MIT Press.

[Margaritis 2003] Margaritis D. (2003). Learning Bayesian Network Model Structure from Data. Ph.D. thesis, School of Computer Science, Carnegie-Mellon University, Pittsburgh, PA. Available as Technical Report CMU-CS-03-153.

[Masuoka 1993] Masuoka R. (1993). Noise Robustness of EBNN learning. Proceedings of the International Joint Conference on Neural Networks, volume 2, pp. 1665-1668. IEEE.

[McCain & Turner 1997] McCain N. and Turner H. (1997). Causal theories of action and change. Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97), pp. 460-465. AAAI Press.

[McCallum and Nigam 1998] McCallum A. and Nigam K. (1998). A Comparison of Event Models for Naive Bayes Text Classication. AAAI-98 Workshop on Learning for Text Categorization, pp. 41-48. AAAI Press.

[McCorduck 2004] McCorduck P. (2004). Machines Who Think: Twenty-Fifth Anniversary Edition. Natick, MA: A. K. Peters, Ltd.

[McGarry & Wermter 2005] McGarry K. and Wermter S. (2005). Training without data: Knowledge Insertion into RBF Neural Networks, International Joint Conference on Artificial Intelligence, pp. 792-798.

[McMillan et al 1992] McMillan C., Mozer M. C., and Smolensky P. (1992). Rule induction through integrated symbolic and subsymbolic processing. In J. E. Moody, S. J. Hanson, and R. P. Lippmann, editors, Advances in Neural Information Processing Systems 4, pages 969–976, San Mateo, CA. Morgan Kaufmann.

[Meek 1995] Meek C. (1995). Strong completeness and faithfulness in Bayesian networks. Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence (UAI-95), pp. 411-418. Morgan Kaufmann.

[Megalooikonomou et al 2005] Megalooikonomou V., Kontos D., DeClaris N., and Cano P. (2005). Utilizing Domain Knowledge in Developing Robust Neural Network Models for Peptide-Allele Binding Prediction. IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology, CIBCB 2005, pp. 364-371. IEEE.

[Menzies & Price 1993] Menzies P. & Price H. (1993). Causation as a secondary quality, British Journal for the Philosophy of Science, 44, pp 187-203.

[Mercer 1909] Mercer J. (1909). Functions of positive and negative type and their connection with the theory of integral equations. Philosophical Transactions of the Royal Society of London, Series A, Containing Papers of a Mathematical or Physical Character, volume 209, pp. 415-446.

[Michalski 1969] Michalski R. S.(1969). On the quasi-minimal solution of the general covering problem. Proceedings of the First International Symposium on Information Processing (pp. 125-128). Bled, Yugoslavia.

[Michalski 1983] Michalski R. (1983). A theory and methodology of inductive learning. Artificial Intelligence, volume 20, number 2, pp. 111-161. Elsvier.

[Michalski et al 1986] Michalski R. S, Mozetic I, Hong J, and Lavrac H.(1986). The multi-purpose incremental learning system AQ15 and its testing application to three medical domains. Proceedings of the Fifth National Conference on AI (pp. 1041-1045). Philadelphia: Morgan Kaufmann.

[Mill 1843] Mill J.S. (1843). A System of Logic, Ratiocinative and Inductive.

[Mingers 1989a] Mingers J.(1989a). An empirical comparison of selection measures for decision tree induction. Machine Learning, 3(4), 319-342.

[Mingers 1989b] Mingers J.(1989b). An empirical comparison of pruning methods for decision tree induction. Machine Learning, 4(2), 227-243.

[Minsky 1961] Minsky M. (1961). Steps toward artificial intelligence. Computers and Thought, pp. 406-450. McGraw-Hill.

[Minsky & Papert 1969] Minsky M., & Papert S. (1969). Perceptrons. Cambridge, MA: MIT Press.

[Minton 1988] Minton S.(1988). Learning search control knowledge: An explanation-based approach. Boston, MA: Kluwer Academic Publishers.

[Mitchell 1977] Mitchell T. M.(1977). Version Spaces: A candidate elimination approach to rule learning. Fifth International Joint Conference on AI, pp 305-310. Cambridge MA: MIT press.

[Mitchell 1997] Mitchell T. M.(1997). Machine Learning. McGraw-Hill Book Co.

[Mitchell & Thrun 1993] Mitchell T. M, & Thrun S. B.(1993). Explanation-based neural network learning for robot control. In Hanson, Cowan, & Giles (Eds.), Advances in neural information processing systems 5, pp. 287-294. San Francisco: Morgan Kaufmann.

[Montazemi & Gupta 1997] Montazemi A. R. and Gupta K. M. (1997). Annals of Operations Research, volume 72, number 0, pp. 51-73. Springer.

[Morik et al 1999] Morik K., Brockhausen P., and Joachims T. (1999). Combining Statistical Learning with a Knowledge-Based Approach - A Case Study in Intensive Care Monitoring. Proceedings of the Sixteenth International Conference on Machine Learning, ICML 1999, pp. 268-277. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA.

[Mozer 1995] Mozer M.(1995). A focused BACKPROPAGATION algorithm for temporal recognition. In Y. Chauvin & D. Rumelhart (Eds.), Backpropagation: Theory, architectures, and applications (pp. 137-169). Hilsdale, NJ: Lawerence Erlbaum Associates.

[Muggleton & Buntine 1988] Muggleton S, & Buntine W.(1988). Machine invention of first order predicates by inverting resolution. Proceedings of the Fifth International Machine Learning Conference (pp. 339-352). Ann Arbor, Michigan: Morgan Kaufmann.

[Muggleton & Feng 1990] Muggleton S, & Feng C. (1990). Efficient induction of logic programs. Proceedings of the First Conference of Algorithmic Learning Theory. Ohmsha, Tokyo.

[Nazeri and Bloedorn 2004] Nazeri Z., Bloedorn E.(2004). Exploiting Available Domain Knowledge to Improve Mining Aviation Safety and Network Security Data. Proceedings ECML/PKDD 2004 Workshop.

[Neal 1996] Neal R. M. (1996). Bayesian Learning for Neural Networks. New York: Springer.

[Neal and Hinton 1998] Neal R. M. and Hinton G. E. (1998). A view of the EM algorithm that justifies incremental, sparse and other variants. Learning in Graphical Models, pp. 355-368. Kluwer Academic Press.

[Neufeld & Kristtorn 2005] Neufeld E. and Kristtorn S. (2005). Whether Non-Correlation Implies Non-Causation. Proceedings of the Eighteenth International Florida Artificial Intelligence Research Society Conference, pp. 772-777. AAAI Press.

[Newell & Simon 1956] Newell A. and Simon H. (1956). The logic theory machine–A complex information processing system. Information Theory, IRE Transactions, volume 2, issue 3, pp. 61-79. IEEE Press.

[Nicholson & Jitnah 1998] Nicholson A. E. and Jitnah N. (1998). Using Mutual Information to determine Relevance in Bayesian Networks. Proceedings of the fifth Pacific Rim International Conference on Artificial Intelligence. Springer.

[Niculescu et al 2006] Niculescu R. S., Mitchell T. M., and Rao R. B. (2006). Bayesian Network Learning with Parameter Constraints. Journal of Machine Learning Research, volume 7, pp. 1357-1383. MIT Press Cambridge, MA, USA.

[Nigam et al 2000] Kamal Nigam and Andrew Mccallum and Sebastian Thrun and Tom Mitchell. Text Classification from Labeled and Unlabeled Documents using EM. Machine Learning, volume 39, issue 2-3, pp. 103-134. Kluwer Academic Publishers Hingham, MA, USA.

[Norton 1994] Norton S. W. (1994) Learning to recognize promoter sequences in E. coli by modelling uncertainty in the training data. In Proceedings of the Twelfth National Conference on Artificial Intelligence, pp 657-663, Seattle, WA.

[Nunez 1991] Nunez M. (1991). The use of background knowledge in decision tree induction. Machine Learning 6(3), 231-250.

[Odajima et al 2008] Odajima K., Hayashi Y., Tianxia G., & Setiono R. (2008). Greedy rule generation from discrete data and its use in neural network rule extraction. Neural Networks, volume 21, pp. 1020-1028. Elsevier Ltd.

[Omohundro 1989] Omohundro, S. (1989). Five Balltree Construction Algorithms. International Computer Science Institute Technical Report tr-89-063.

[Opitz and Shavlik] Opitz D. W. and Shavlik J. W. (1995). Dynamically adding symbolically meaningful nodes to knowledge-based neural networks, Knowledge-Based Systems, vol. 8, no. 6, pp. 301-311.

[Opper & Haussler 1991] Opper M., & Haussler D. (1991). Generalization performance of Bayes optimal prediction algorithm for learning a perceptron.. Physical Review Letters, 66, 2677-2681.

[O'Reilly & Oppacher 1994] O'Reilly U-M, & Oppacher R. (1994). Program search with hierarchical variable length representation: Genetic programming, simulated annealing, and hill climbing. In Y. Davidor et al. (Eds). Parallel problem solving from nature–PPSN III (Vol 866) (Lecture notes in computer science). Springer Verlag.

[Pagallo & Haussler 1990] Pagallo G., & Haussler D. (1990). Boolean feature discovery in empirical learning. Machine Learning, volume 5, number 1, pp. 71-99. Kluwer Academic Publishers Hingham, MA, USA.

[Panait & Luke 2005] Panait L. and Luke S. (2005). Cooperative Multi-Agent Learning: The State of the Art. Autonomous Agents and Multi-Agent Systems, volume 11, issue 3, pp. 387-434. Kluwer Academic Publishers Hingham, MA, USA.

[Pappas & Gillies 2002] Pappas A., and Gillies D.F. (2002). A New Measure for the Accuracy of a Bayesian Network. MICAI 2002: Advances in Artificial Intelligence, pp. 411-419. Springer Berlin / Heidelberg.

[Parekh and Honavar 1998] Parekh R. and Honavar V. (1998). Constructive theory refinement in knowledge based neural networks. In Proceedings of the International Joint Conference on Neural Networks, pp. 2318-2323. IEEE.

[Park et al 2007] Park C., Koo J., Kim S., Sohn I. and Lee J. W. (2007). Classification of gene functions using support vector machine for time-course gene expression data. Computational Statistics & Data Analysis Volume 52, Issue 5, pp 2578-2587. Elsevier Science Publishers B. V. Amsterdam, The Netherlands.

[Passerini et al 2006] Passerini A., Frasconi P., and De Raedt L. (2006). Kernels on Prolog Proof Trees: Statistical Learning in the ILP Setting. The Journal of Machine Learning Research, volume 7, pp. 307-342. MIT Press Cambridge, MA, USA.

[Pavlov 1927] Pavlov I.P. (1927). Conditioned reflexes: An investigation of the physiological activity of the cerebral cortex. Translated by G. V. Anrep published by Oxford University Press. Reprinted 1960 by Dover Publications, New York.

[Pawlowski et al 2009] Pawlowski M., Paterek T., Kaszlikowski D., Scarani V., Winter A., and Zukowski M. (2009). A new physical principle: Information Causality. Nature, volume 461, pp. 1101-1104. Nature Publishing Group.

[Pazzani 1989] Pazzani M. (1989). Explanation-based learning with weak domain theories. Proceedings of the Sixth International Workshop on Machine Learning (pp. 72– 74). Ithaca, NY: Morgan Kaufmann.

[Pazzani et al 1991] Pazzani M. J., Brunk C. A, & Silverstein G. (1991). A knowledge-intensive approach to learning relational concepts. Proceedings of the Eighth International Workshop on Machine Learning, (pp. 432-436). San Mateo, CA; Morgan Kaufmann.

[Pazzani & Kibler 1992] Pazzani M. J., & Kibler D. (1992). The utility of knowledge in inductive learning. Machine Learning, 9(1), 57-94.

[Pazzani 1993] Pazzani M. (1993). Learning Causal Patterns: Making a Transition from data-Driven to Theory-Driven Learning, Machine Learning 11, pp 173-194. KluwerAcademic Publishers AAAI/MIT Press.

[Pearl 1988] Pearl J. (1988). Probabilistic reasoning in intelligent systems: networks of plausible inference. San Mateo, California: Morgan Kaufmann.

[Pearl 1993] Pearl J. (1993). Comment: Graphical Models, Causality and Intervention. Statistical Science, volume 8, number 3, pp. 266-269. Institute of Mathematical Statistics.

[Pearl 1994] Pearl J. (1994). Causal Diagrams for Empirical Research. Report R218-B, Cognitive Systems Laboratory, Computer Science Department, University of California, Los Angeles, USA.

[Pearl 1995] Pearl J. (1995). Causal diagrams for empirical research. Biometrika, volume 82, number 4, pp. 669-688. Biometrika Trust.

[Pearl 2000] Pearl J. (2000). Causality: models, reasoning and inference. Cambridge University Press.

[Pearl and Verma 1991] Pearl J. and Verma T. (1991). A Theory of Inferred Causation. Proceedings of the Second International Conference on the Principles of Knowledge Representation and Reasoning, pp. 441-52. Morgan Kaufmann.

[Peirce 1877] Peirce C. S. (1877). The Fixation of Belief. Popular Science Monthly 12, pp 1-15.

[Peirce 1958] Peirce C. S. (1958). The Collected Papers of Charles Sanders Peirce. Editor Arthur W. Burks, Volumes 7 and 8. Harvard University Press, Cambridge, MA

[Peña et al 2005] Peña J.M., Björkegren J., and Tegnér J. (2005). Learning dynamic Bayesian network models via cross-validation. Pattern Recognition Letters, volume 26 , issue 14, pp. 2295-2308. Elsevier Science Inc. New York, NY, USA.

[Penrose & Percival 1962] Penrose O. and Percival I.C. (1962). The Direction of Time. Proceedings of the Physical Society, Volume 79, Number 3, pp. 605-616. IOP Publishing.

[Perrier et al 2008] Perrier E., Imoto S., and Miyano S. (2008). Finding Optimal Bayesian Network Given a Super-Structure. Journal of Machine Learning Research, volume 9, pp. 2251-2286. Microtome Publishing.

[Pomerleau 1993] Pomerleau D. A. (1993). Knowledge-based training of artificial neural networks for autonomous robot driving. In J. Connell & S. Mahadevan (Eds.), Robot Learning (pp. 19-43). Boston: Kluwer Academic Publishers.

[Popper 1934] Popper K. R. (1934). The Logic of Scientific Discovery, with new appendices of 1959, London: Routledge, 1999.

[Popper 1957] Popper K. R. (1957). The Propensity Interpretation of the Calculus of Probability and the Quantum Theory. Proceedings of the Ninth Symposium of the Colston Research Society, pp. 65-70. Butterworth Scientific Publications.

[Popper 1959] Popper K. R. (1959). The Propensity Interpretation of Probability. The British Journal for the Philosophy of Science, volume 10, number 37, pp. 25-42. Oxford University Press.

[Porter & Bareiss 1986] Porter B. W. & Bareiss E. R. (1986). PROTOS: An experiment in knowledge acquisition for heuristic classification tasks. In, Proceedings of the First International Meeting on Advances in Learning (IMAL), pp. 159-174. Springer-Verlag.

[Price 1991] Price H. (1991). Agency and probabilistic causality, British Journal for the Philosophy of Science, volume 42, number 2, pp. 157-176. Oxford University Press.

[Putnam 1981] Putnam H. (1981). Reason, Truth and History. Cambridge University Press, Cambridge, England.

[Quinlan 1986] Quinlan J. R. (1986). Induction of decision trees. Machine Learning, volume 1, issue 1, pp. 81-106. Kluwer Academic Publishers.

[Quinlan 1987] Quinlan J. R. (1987). Rule induction with statistical data—a comparison with multiple regression. Journal of the Operational Research Society, 38, 347-352.

[Quinlan 1990] Quinlan J. R. (1990). Learning logical definition from relations. Machine Learning, 5, 239-266.

[Quinlan 1993] Quinlan J. R. (1993). C4.5: Programs for Machine Learning. San Mateo, CA: Morgan Kaufmann.

[Qunilan & Cameron-Jones 1993] Quinlan J.R., and Cameron-Jones R. M. (1993). FOIL: A Midterm Report. Proceedings of the European Conference on Machine Learning (ECML-93), pp. 3-20. Springer.

[Ramoni and Sebastiani 1997] Ramoni M. and Sebastiani P. (1997). Discovering Bayesian networks in incomplete databases. Technical report KMI-TR-46, Knowledge Media Institute, The Open University.

[Ramsey 1926] Ramsey F. P. (1926). Truth and Probability. The Foundations of Mathematics and other Logical Essays, edited by Braithwaite, R. B., chapter 7, pp. 156-198. Harcourt Brace & Co.

[Ranzato et al 2007] Ranzato M., Huang F.J., Boureau Y., and LeCun Y. (2007). Unsupervised Learning of Invariant Feature Hierarchies with Applications to Object Recognition. Proceedings of Computer Vision and Pattern Recognition Conference (CVPR 2007). IEEE press.

[Rebane & Pearl 1987] Rebane G. and Pearl J. (1987). The Recovery of Causal Poly-Trees From Statistical Data. Proceedings of the Uncertainty in Artificial Intelligence Third Annual Conference on Uncertainty in Artificial Intelligence (UAI-87), pp. 222-228. Elsevier Science.

[Reichenbach 1949] Reichenbach H. (1949). The Theory of Probability. University of California Press.

[Reichenback 1956] Reichenbach, H. (1956). The Direction of Time, Berkeley, University of Los Angeles Press.

[Richards et al 2005] Richards G., Brazier K.J., and Wang W. (2005). A Fast Wrapper Method for Feature Subset Selection. Proceedings of the Artificial Intelligence and Applications, AIA2005, pp. 54-59. ACTA Press.

[Richardson & Domingos 2006] Richardson M., & Domingos P. (2006). Markov logic networks. Machine Learning, volume 62, pp. 107-136. Springer Netherlands.

[Richardson 2003] Richardson T. (2003). Markov Properties for Acyclic Directed Mixed Graphs. Scandinavian Journal of Statistics, volume 30, issue 1, pp. 145-157. Blackwell publishing.

[Riedmiller & Braun 1993] Riedmiller M., and Braun H. (1993). “A direct adaptive method for faster backpropagation learning: The RPROP algorithm”. IEEE International Conference on Neural Networks, pp. 586-591. IEEE.

[Rimer and Martinez 2006] Rimer M., and Martinez T. (2006). Classification-based objective functions. Machine Learning, volume 63, issue 2, pp. 183-205. Kluwer Academic Publishers Hingham, MA, USA.

[Rissanen 1978] Rissanen J. (1978). Modeling by shortest data description. Automatica, volume 14, pp. 465-471. Elsevier B.V.

[Rissanen 1987] Rissanen J. (1987). Stochastic complexity (with discussion). Journal of the Royal Statistical Society, Series B, volume 49, number 3, pp. 223-239 and 253-265.

[Robinson 1965] Robinson J. A. (1965). A machine-oriented logic based on the resolution principle. Journal of the ACM, 12(1), 23-41.

[Roeper & Leblanc 1999] Roeper P. and Leblanc H. (1999). Probability Theory and Probability Logic. Toronto: University of Toronto Press.

[Rosenblatt 1957] Rosenblatt F. (1957). The perceptron: A perceiving and recognizing automaton (project PARA). Technical Report 85-460-1, Cornell Aeronautical Laboratory.

[Roth and Fischer 2008] Roth V., Fischer B. (2008). The Group-Lasso for Generalized Linear Models: Uniqueness of Solutions and Efficient Algorithms. Proceedings of the twenty fifth international conference on Machine learning (ICML 2008), pp 848-855. ACM New York, NY, USA.

[Rubin 1989] Rubin H.(1989). Discussion of "The Logic of Influence Diagrams" by Pearl et al. Influence Diagrams, Belief Networks and Decision Analysis, pp 83-85. John Wiley and Sons, Ltd., Sussex, England.

[Rückert and Kramer 2006] Rückert U, and Kramer S. (2006). A statistical approach to rule learning. Proceedings of the Twenty-Third International Conference (ICML 2006), pp. 785-792. ACM Press.

[Rückert & De Raedt 2008] Rückert U, and De Raedt L. (2008). An experimental evaluation of simplicity in rule learning. Artificial Intelligence, volume 172, issue 1, pp. 19-28. Elsevier Science Publishers Ltd. Essex, UK.

[Ruffo 2000] Ruffo G.(2000). Learning Single and Multiple Instance Decision Trees for Computer Security Applications. Doctoral dissertation, Department of Computer Science, University of Torino.

[Rumelhart et al 1986] Rumelhart D. E., Hinton G. E., & Williams R. J. (1986). Learning representations by back-propagating errors. Nature 323, pp. 533-536. Nature Publishing Group.

[Rusakov & Geiger 2005] Rusakov D. and Geiger D. (2005). Asymptotic Model Selection for Naive Bayesian Networks. Journal of Machine Learning Research, volume 6, pp. 1-35. Microtome Publishing.

[Russell 1913] Russell B. (1913). On the Notion of Cause. Proceedings of the Aristotelian Society, volume 13, pp. 1-26.

[Russel et al 1995] Russel S, Binder J, Koller D, Kanazawa K.(1995). Local learning in probabilistic networks with hidden variables. Proceedings of the 14th International Joint Conference on Artificial Intelligence, Montreal. San Francisco: Morgan Kaufmann.

[Russell & Norvig 1995] Russell S. J. and Norvig P. (1995). Artificial Intelligence: A Modern Approach. Prentice Hall.

[Salakhutdinov et al 2003] Salakhutdinov R., Roweis S. and Ghahramani Z. (2003). Optimization with EM and Expectation-Conjugate-Gradient. Proceedings of the twentieth International Conference on Machine Learning (ICML-2003), pp. 672-679. AAAI Press.

[Salmon 1998] Salmon W. C.(1998). Causality and explanation. Oxford University Press.

[Salzberg 1995] Salzberg S.(1995). Locating protein coding regions in human DNA using a decision tree algorithm. Journal of Computational Biology 2 (3), pp 473-485.

[Samet 1984] Samet H.(1984). The quadtree and related hierarchical data structures. ACM Computing Surveys, volume 16 (2), pp. 187-260. ACM Press.

[Sánchez-Maroño et al 2005] Sánchez-Maroño N., and Alonso-Betanzos A., and Castillo E.(2005). A New Wrapper Method for Feature Subset Selection. Proceedings of the European Symposium on Artificial Neural Networks 2005, ESANN 2005, pp 515-520.

[Sato 1995] Sato T. (1995). A statistical learning method for logic programs with distribution semantics. In Proceedings of the 12th international conference on logic programming (ICLP-1995) (pp. 715–729).

[Sato et al 2008] Sato T., Zhou N.-F., Kameya Y., & Izumi Y. (2008). Prism user's manual (version 1.11.2). `http://sato-www.cs.titech.ac.jp/prism/`.

[Savage 1954] Savage L. J. (1954). The Foundations of Statistics. John Wiley and Sons.

[Schank 1982] Schank R. C. (1982). Dynamic Memory: A Theory of Learning in Computers and People. Cambridge University Press.

[Schapire 1990] Schapire R.(1990). The Strength of Weak Learnability. Machine Learning, volume 5(2), pp 197-227. Springer Netherlands.

[Schapire 1997] Schapire R. (1997). Using output codes to boost multiclass learning problems. Proceedings of the Fourteenth International Conference on Machine Learning (ICML 1997), pp 313-321. Morgan Kaufmann.

[Schapire & Singer 1999] Schapire R. and Singer Y. (1999). Improved Boosting Algorithms Using Confidence-rated Predictions. Proceedings of the Eleventh Annual Conference on computational Learning Theory, pp 297-336. Kluwer Academic Publishers Hingham, MA, USA.

[Scheines et al 1994] Scheines R., Spirtes P., Glymour C., and Meek C. (1994). TETRAD II: Tools for Discovery. Lawrence Erlbaum Associates, Hillsdale, NJ.

[Schervish et al 2000] Schervish M. J., Seidenfeld T., and Kadane J. B. (2000). How sets of coherent probabilities may serve as models for degrees of incoherence. Journal of Uncertainty, Fuzziness, and Knowledge-based Systems, volume 8, issue 3, pp. 347-356. World Scientific Publishing Co.

[Scholkopf & Smola 2002] Schölkopf B., & Smola A. (2002). Learning with kernels. Cambridge, MA: MIT Press.

[Schwarz 1978] Schwarz G.E. (1978). Estimating the Dimension of a Model. Annals of Statistics, volume 6, issue 2, pp. 461-464. Institute of Mathematical Statistics.

[Seidenfeld et al 1989] Seidenfeld T., Kadane J. and Schervish M. (1989). On the Shared Preferences of Two Bayesian Decision Makers. Journal of Philosophy, volume 86, issue 5, pp. 225–244. Journal of Philosophy, Inc.

[Settles 2009] Settles B. (2009). Active Learning Literature Survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison. This is an updated online document which can be found at `http://pages.cs.wisc.edu/~bsettles/active-learning/`.

[Shah 2007] Shah M. (2007). Sample Compression Bounds for Decision Trees. Proceedings of the twenty fourth international conference on Machine learning, (ICML 2007), pp 799-806. ACM New York, NY, USA.

[Shannon & Weaver 1949] Shannon C. E, & Weaver W. (1949). The mathematical theory of communication. Urbana: University of Illinois Press.

[Sharma & Sleeman 1988] Sharma S. and Sleeman D. H. (1988). REFINER: A Case-Based Differential Diagnosis Aide for Knowledge Acquisition and Knowledge Refinement. Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, ECML, pp. 201-210. Pitman.

[Shavlik & Towell 1989] Shavlik J, & Towell G. (1989). An approach to combining explanation-based and neural learning algorithms. Connection Science, volume 1(3), 233-255.

[Shortliffe & Buchanan 1975] Shortliffe E. H. and Buchanan B. G. (1975). A model of inexact reasoning in medicine. Mathematical Biosciences, volume 23, issues 3-4, pp. 351-379. Elsevier B.V.

[Silva & Ghahramani 2009] Silva R. and Ghahramani Z. (2009). The Hidden Life of Latent Variables: Bayesian Learning with Mixed Graph Models. Journal of Machine Learning Research, volume 10, pp. 1187-1238. Microtome Publishing.

[Silverstein et al 2000] Silverstein C., Brin S., Motwani R., and Ullman J. D. (2000). Scalable Techniques for Mining Causal Structures. Data Mining and Knowledge Discovery 4, pp 163-192, Kluwer Academic Publishers, The Netherlands.

[Simard et al 1992] Simard P. S, Victorri B, LeCun Y, & Denker J. (1992). Tangent prop—A formalism for specifying selected invariances in an adaptive network. In Moody et al. (Eds.) Advances in Neural Information Processing Systems 4 (pp. 895-903). San Francisco: Morgan Kaufmann.

[Simon 1953] Simon H. (1953). Causal Ordering and Identifiability. Studies in Econometric Method, pp. 49-74. Wiley, New York, USA.

[Skyrms 1984] Skyrms B. (1984). Pragmatics and Empiricism. New Haven: Yale University Press.

[Simpson 1951] Simpson E. H. (1951). The Interpretation of Interaction in Contingency Tables. Journal of the Royal Statistical Society, Series B, volume 13, pp. 238-241.

[Solomonoff 1964a] Solomonoff R. (1964). A Formal Theory of Inductive Inference, Part I. Information and Control, volume 7, number 1, pp. 1-22. Elsevier Inc.

[Solomonoff 1964b] Solomonoff R. (1964). A Formal Theory of Inductive Inference, Part II. Information and Control, volume 7, number 2, pp. 224-254. Elsevier Inc.

[Spiegelhalter et al 1989] Spiegelhalter D. J., Franklin R. C. G. and Bull K. (1989). Assessment, Criticism, and Improvement of Imprecise Probabilities for a Medical Expert System. Proceedings of the Fifth Annual Conference on Uncertainty in Artificial Intelligence, pp. 285-294. North-Holland Publishing Co.

[Spirtes & Glymour 1990] Spirtes P., & Glymour C. (1990). Causal Structure among Measured Variables preserved with Unmeasured Variables. Carnegie Mellon University, Department of Philosophy Report CMU-PHIL-14.

[Spirtes et al 1991] Spirtes P., Glymour C., and Scheines R. (1991). An Algorithm for Fast Recovery of Sparse Causal Graphs. Social Science Computer Review, volume 9, pp. 62-72. Sage Publications.

[Spirtes et al 1993] Spirtes P., Glymour C., and Scheines R. (1993). Causation, Prediction and Search. Springer Lecture Notes in Statistics, no 81, Springer-Verlag, New York.

[Spirtes et al 1995] Spirtes P., Meek C., and Richardson T. (1995). Causal Inference in the Presence of Latent Variables and Selection Bias. In Proceedings of Eleventh Conference on Uncertainty in Artificial Intelligence, pp. 499-506. Morgan Kaufmann.

[Spohn 1986] Spohn W. (1986). The Representation of Popper Measures. Topoi, volume 5, number 1, pp. 69-74. Springer.

[Srinivasan et al 1995] Srinivasan A, Muggleton S, & King R. D.(1995). Comparing the use of background knowledge by inductive logic programming systems (PRG Technical report PRG-TR-9-95). Oxford University Computing Laboratory.

[Srinivasan, King & Bain 2003] Srinivasan A., King R. D., and Bain M. E. (2003). An Empirical Study of the Use of Relevance Information in Inductive Logic Programming. The Journal of Machine Learning Research, volume 4, pp. 369-383. MIT Press Cambridge, MA, USA.

[Strawson 1950] Strawson P.F. (1950). On Referring. Mind, New Series, volume 59, number 235, pp. 320-344. Oxford University Press.

[Strube 1992] Strube G. (1992). The role of cognititve science in knowledge engineering. Contemporary Knowledge Engineering and Cognition Lecture Notes in Computer Science, volume 622, pp. 159-174. Springer.

[Sulzmann et al 2007] Sulzmann J.-N., Fürnkranz J., and Hüllermeier E. (2007). On Pairwise Naive Bayes Classifiers. Machine Learning: ECML 2007, pp. 371-381. Springer-Verlag Berlin, Heidelberg.

[Sun & DeJong 2005] Sun Q. and DeJong G. (2005). Explanation-Augmented SVM: an approach to incorporating domain knowledge into SVM learning. Proceedings of the twenty second international conference on Machine learning, ICML 2005, pp. 864-871. ACM New York, NY, USA.

[Sun et al 2007] Sun X., Janzing D., Schölkopf B., Fukumizu K.(2007). A Kernel-based Causal Learning Algorithm. Proceedings of the 24 Annual International Conference on Machine Learning (ICML 2007), pp 855-862. ACM Press, New York, NY, USA.

[Suppes 1970] Suppes P. (1970). A Probabilistic Theory of Causality. Amsterdam: North-Holland Publishing Co.

[Sutskever & Hinton 2007] Sutskever I. and Hinton G.E. (2007). Learning multilevel distributed representations for high-dimensional sequences. Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics, volume 2, pp. 548-555. Omnipress.

[Sutton 1988] Sutton R. S. (1988). Learning to predict by the methods of temporal differences. Machine Learning, volume 3, number 1, pp. 9-44. Springer Netherlands.

[Sutton & Tanner 2004] Sutton R. S. and Tanner B. (2004). Temporal-Difference Networks. Advances in Neural Information Processing Systems 17, pp. 1377-1384. Cambridge, MA: MIT Press.

[Suzuki et al 2001] Suzuki E., Gotoh M., and Choki Y. (2001). Bloomy Decision Tree for Multi-objective Classification. Principles of Data Mining and Knowledge Discovery, pp. 436-447. Springer Berlin, Germany.

[Suzuki 1996] Suzuki J. (1996). Learning Bayesian belief networks based on the MDL principle: An efficient algorithm using the branch and bound technique. Proceedings of the International Conference on Machine Learning, ICML 1996, pp. 462-470. Morgan Kaufmann.

[Tan 1993] Tan M. (1993). Cost sensitive learning of classification knowledge and its application in robotics. Machine Learning, volume 13, issue 1, pp. 1-33.

[Tan and Schlimmer 1990] Tan M., & Schlimmer J. C. (1990). Two case studies in cost-sensitive concept acquisition. Proceedings of the AAAI-90.

[Taylor et al 2007] Taylor G.W., Hinton G.E. and Roweis S.T. (2007). Modeling human motion using binary latent variables. Advances in Neural Information Processing Systems, volume 19, pp. 1345-1352. MIT Press, Cambridge, MA

[Teller & Veloso 2000] Teller A. and Veloso M. (2000). Efficient Learning Through Evolution: Neural Programming and Internal Reinforcement. Proceedings of the Seventeenth International Conference on Machine Learning, ICML 2000, pp. 959-966. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA.

[Thrun et al 1991] Thrun S. B., Bala J. , Bloedorn E., Bratko I., Cestnik B., Cheng J., De Jong K., Dzeroski S., Fahlman S. E., Fisher D., Hamann R., Kaufman K., Keller S., Kononenko I., Kreuziger J., Michalski R.S., Mitchell T., Pachowicz P., Reich Y., Vafaie H., Van De Welde W., Wenzel W., Wnek J., Zhang J. (1991). The monk's problems: A performance comparison of different learning algorithms. Technical Report CMU-CS-91-197, Carnegie Mellon University.

[Thrun 1993] Thrun S. B. (1993). Extracting Provably Correct Rules From Artificial Neural Networks, Technical Report IAI-TR-93-5, Institut fur Informatik III, Universität Bonn.

[Thrun 1996] Thrun S. B. (1996). Explanation-based neural network learning: A lifelong learning approach. Boston: Kluwer Academic Publishers.

[Tibshirani 1994] Tibshirani R. (1994). Regression Shrinkage and Selection Via the Lasso. Journal of the Royal Statistical Society, Series B, volume 58, pp. 267-288. Royal Statistical Society, England.

[Tibshirani and Hastie 2007] Tibshirani R. and Hastie T. (2007). Margin Trees for High-dimensional Classification. The Journal of Machine Learning Research, volume 8, pp. 637-652. MIT Press Cambridge, MA, USA.

[Tighe & Tawfik 2008] Tighe C. A. and Tawfik A. Y. (2008). Using causal knowledge to guide retrieval and adaptation in case-based reasoning about dynamic processes. International Journal of Knowledge-based and Intelligent Engineering Systems, volume 12, issue 4, pp. 271-281. IOS Press.

[Tinbergen 1940] Tinbergen J. (1940). Reply to Keynes, The Economics Journal 50, pp. 141-146.

[Towell et al 1990] Towell G.G., Shavlik J.W., Noordewier M.O. (1990). Refinement of Approximate Domain Theories by Knowledge-based Neural Networks. Proceedings of the eight national conference on Artificial Intelligence, volume 2, pp. 861-866. MIT Press.

[Tsamardinos et al 2006] Tsamardinos I., Brown L.E., and Aliferi C.F. (2006). The max-min hill-climbing Bayesian network structure learning algorithm. Machine Learning, volume 65, pp. 31–78. Springer Netherlands.

[Turing 1937] Turing A. M. (1937). On Computable Numbers, with an Application to the Entscheidungsproblem. Proceedings of the London Mathematical Society, series 2, volume 42, issue 1, pp. 230-265. Oxford University Press.

[Tuv et al 2009] Tuv E., Borisov A., Runger G., and Torkkola K. (2009). Feature Selection with Ensembles, Artificial Variables, and Redundancy Elimination. Journal of Machine Learning Research, volume 10, pp. 1341-1366. Microtome Publishing.

[Tversky & Kahneman 1974] Tversky A. and Kahneman D. (1974). Judgment under Uncertainty: Heuristics and Biases. Science, volume 185, number 4157, pp. 1124-1131. American Association for the Advancement of Science, New York, USA.

[Utgoff 1986] Utgoff P.E. (1986). Machine learning of inductive bias. Kluwer, B.V. Deventer, The Netherlands.

[Valiant 1984] Valiant L. (1984). A Theory of the Learnable. Communications of the ACM, volume 27, number 11, pp 1134-1142. ACM New York, NY, USA.

[Vapnik 1982] Vapnik V. (1982). Estimation of Dependences Based on Empirical Data. Information Science and Statistics series, Springer-Verlag.

[Vapnik 1998] Vapnik V. (1998). Statistical Learning Theory. Wiley, New York.

[Vapnik & Chervonenkis 1971] Vapnik V. and Chervonenkis A. (1971). On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities. Theory Probability and its Applications Volume 16, Issue 2, pp. 264-280. Society for Industrial and Applied Mathematics.

[Venn 1866] Venn J. (1866). The Logic of Chance. Macmillan and co.

[Verma and Pearl 1990] Verma T., Pearl J. (1990). Equivalence and synthesis of causal models. Proceedings of the Sixth Annual Conference on Uncertainty in Artificial Intelligence, pp. 255-270. Elsevier Science Inc. New York, NY, USA.

[von Mises 1957] von Mises R. (1957). Probability, Statistics and Truth. George Allen & Unwin Limited.

[Waldinger 1977] Waldinger R. (1977). Achieving several goals simultaneously. In E. Elcock & D. Michie (Eds.), Machine Intelligence, 8. London: Ellis Horwood Ltd.

[Wall & Cunningham 2000] Wall R., Cunningham P. (2000). Exploring the Potential for Rule Extraction from Ensembles of Neural Networks. Proceedings of the eleventh Irish Conference on Artificial Intelligence & Cognitive Science (AICS 2000), pp. 52-68. World Scientific Publishing Co.

[Webb et al 2005] Webb G. I., Boughton J. R. and Wang Z. (2005). Not So Naive Bayes: Aggregating One-Dependence Estimators. Machine Learning, volume 58, number 1, pp. 5-24. Kluwer Academic Publishers.

[Weinberger et al 2006] Weinberger K., Blitzer J., & Saul L. (2006). Distance Metric Learning for Large Margin Nearest Neighbor Classification. In Advances in neural information processing systems, volume 18, pp 1473-1480. Cambridge, MA: MIT Press.

[Weinberger & Saul 2008] Weinberger K. and Saul L. (2008). Fast Solvers and Efficient Implementations for Distance Metric Learning. Proceedings of the twenty fifth Annual International Conference on Machine Learning (ICML 2008), pp 1160-1167. Omnipress.

[Weisberg 1980] Weisberg, S. (1980). Applied Linear Regression. Wiley, New York.

[Weiss and Indurkhya 2000] Weiss S. and Indurkhya N. (2000). Lightweight Rule Induction. Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), pp. 1135-1142. Morgan Kaufmann.

[Whiteson & Stone 2006] Whiteson S. and Stone P. (2006). Evolutionary Function Approximation for Reinforcement Learning. Journal of Machine Learning Research, volume 7, pp. 877-917. MIT Press Cambridge, MA, USA.

[Williams & Zipser 1995] Williams R, & Zipser D. (1995). Gradient-based learning algorithms for recurrent networks and their computational complexity. In Y. Chauvin & D. Rumelhart (Eds.), Backpropagation: Theory, architectures, and applications (pp. 433-486). Hilsdale, NJ: Lawerence Erlbaum Associates.

[Williamson 2004] Williamson J. (2004). Bayesian nets and causality: philosophical and computational foundations. Oxford: larendon Press.

[Williamson 2006] Williamson J. (2006). Causality. In Gabbay D. and Guenthner F., editors, Handbook of Philosophical Logic, volume 14, Springer.

[Wilson 1970] Wilson P.H.(1970). Learning structural descriptions from examples. (Ph.D. dissertation). MIT Technical Report AI-TR-231.

[Winkler 1996] Winkler R. L. (1996). Scoring Rules and the Evaluation of Probabilities. Test, volume 5, number 1, pp. 1-60. Springer.

[Winograd 1971] Winograd T. (1971). Procedures as a Representation for Data in a Computer Program for Understanding Natural Language. MIT AI Technical Report 235. MIT Press.

[Wojnarski 2007] Wojnarski M. (2007). Nondeterministic Discretization of Weights Improves Accuracy of Neural Networks. Proceedings of the eighteenth European conference on Machine Learning, ECML 2007, pp. 765-772. Springer-Verlag Berlin, Heidelberg.

[Wolfe Haghight & Klein 2008] Wolfe J., Haghighi A., and Klein D. (2008). Fully Distributed EM for Very Large Datasets. Proceedings of the twenty fifth International Conference on Machine learning (ICML 2008), pp. 1184-1191. ACM New York, NY, USA.

[Wolfram 2002] Wolfram S. (2002). A New Kind of Science. Wolfram Media Inc.

[Wong & Xiang 1994] Wong S.K.M. and Xiang Y. (1994). Construction of a Markov Network from Data for Probabilistic Inference. Proceedings of the Third International Conference Rough Sets and Soft Computing, pp. 562-569.

[Wright 1921] Wright S. (1921). Correlation and Causation. Journal of Agricultural Research, volume 20, number 7, pp. 557-585. United States Department of Agriculture, Washington D.C., USA.

[Wu et al 1999] Wu D., Bennett K., Christiani N, and Shawe-Taylor J. (1999). Large Margin Decision Trees for Induction and Transduction. Proceedings of the Sixteenth International Conference on Machine Learning (ICML99), pp 474-483.

[Xing et al 2003] Xing E., Ng A., Jordan M., & Russell S. (2003). Distance Metric Learning, with Application to Clustering with Side-Information. Advances in Neural Information Processing Systems, volume 15, pp 505-512. Cambridge, MA: MIT Press.

[Yang and Webb 2009] Yang Y. and Webb G. I. (2009). Discretization for naive-Bayes learning: managing discretization bias and variance. Machine Learning, volume 74, issue 1, pp. 39-74. Kluwer Academic Publishers Hingham, MA, USA.

[Yehezkel & Lerner 2009] Yehezkel R. and Lerner B. (2009). Bayesian Network Structure Learning by Recursive Autonomy Identification. Journal of Machine Learning Research, volume 10, pp. 1527-1570. Microtome Publishing.

[Yin Wang & Wu 2004] Yin D.-S., Wang G.-Y., Wu Y. (2004). A Self-Learning Algorithm for Decision Tree Pre-Pruning. Proceedings of 2004 International Conference on Machine Learning and Cybernetics, volume 4, pp 2140- 2145. IEEE.

[Young 1987] Young J.O. (1987). Global Anti-Realism. Philosophy and Phenomenological Research, volume 47, number 4, pp. 641-647. International Phenomenological Society, USA.

[Yuan and Lin 2006] Yuan M. and Lin Y. (2006). Model selection and estimation in regression with grouped variables. Journal of the Royal Statistical Society, Series B, volume 68, pp 49-67. Wiley, New York.

[Zhang et al 2005] Zhang H., Jiang L., and Su J. (2005). Augmenting Naive Bayes for Ranking. Proceedings of the Twenty Second International Conference on Machine learning, pp. 1020-1027. ACM New York, NY, USA .

[Zhang and Oles 2000] Zhang T., and Oles F. (2000). A probability analysis on the value of unlabeled data for classification problems. Proceedings of the Seventeenth International Conference on Machine Learning (ICML2000), pp 1191-1198. Morgan Kaufmann.

[Zhang, Baral & Kim 2005] Zhang X., Baral C., Kim, S. (2005). An Algorithm to Learn Causal Relations Between Genes from Steady State Data: Simulation and Its Application to Melanoma Dataset. Lecture Notes in Computer Science, number 3581, pp. 524-534. Springer-Verlag, Germany.

[Zhang et al 2008] Zhang Z., Dai B. T., and Tung A. K. H. (2008). Estimating Local Optimums in EM Algorithm over Gaussian Mixture Model. Proceedings of the Twenty Fifth International Conference on Machine Learning, ICML 2008, pp. 1240-1247. ACM New York, NY, USA.

[Zheng & Webb 2000] Zheng Z. & Webb G. I. (2000). Lazy learning of Bayesian Rules. Machine Learning, volume 41, issue 1, pp. 53–84. Kluwer Academic Publishers.

[Zhong et al 2008] Zhong M., Georgiopoulos M., and Anagnostopoulos G. (2008). A k-norm pruning algorithm for decision tree classifiers based on error rate estimation. Machine Learning, volume 71, number 1, pp. 55-88. Springer Netherlands.

[Zhou et al 2003] Zhou Z.-H., Jiang Y., and Chen S.-F. (2003). Extracting symbolic rules from trained neural network ensembles. AI Communications, volume 16, issue 1, pp. 3-15. IOS Press Amsterdam, The Netherlands.

[Zhu et al 2005] Zhu X., Kandola J., Ghahramani Z., and Lafferty J. (2005). Nonparametric transforms of graph kernels for semi-supervised learning. Advances in Neural Information Processing Systems 17, pp 1641-1648. MIT Press, Cambridge, MA, USA.

[Zhu 2008] Zhu X. (2008). Semi-Supervised Learning Literature Survey. Computer Sciences Technical Report 1530, University of Wisconsin, Madison. This is an updated online document to be found at `http://pages.cs.wisc.edu/~jerryzhu/research/ssl/semireview.html`.

[Zyl and Omlin 2001] Zyl J. van and Omlin C. W. (2001). Knowledge-Based Neural Networks for Modelling Time Series, Lecture Notes in Computer Science, 2085, pp 579-586.